IT*Script*Net® Batch Plusä

Version 2.4

Create Powerful Data Collection
Solutions - Simply and Easily

User Guide

Printed in USA.

Table of Contents

SOFTWARE LICENSE AGREEMENT

READ THIS BEFORE USING THE NOTED PROGRAMS

Thank you for selecting ITScriptNet® from Z-Space Technologies, Inc. ("Z-Space"). Please read the following License Agreement below before registering the serial number. If you do not accept these terms, return the product unregistered with proof of purchase to the point of purchase for a complete refund. Only if you accept these terms should you register the software.

The enclosed copy of ITScriptNet® is never sold. It is licensed by Z-Space to the original customer and to any subsequent licensee of his or her for use in accordance with the terms set forth below. BY REGISTERING THIS SOFTWARE YOU ARE INDICATING ACCEPTANCE OF THESE TERMS. Otherwise, you may return the software and User Guide within ten (10) days to the place where you obtained it for a full refund. Under the terms of this license agreement:

YOU MAY:

For PC-Based Licenses:

1. *Load and use* the software on any computer as long as it is used on only one computer by one user at a time. The software serial number can only be registered once on a single computer. The software cannot be shared over a network. If more than one computer requires the use of the software, then additional license fees will be required for each computer.
2. *Communicate* with the Remote Host Server (ITScriptNet® OMNI™ Server) residing on the host computer with only the number of terminals as there are terminal licenses registered on the host computer. Additional terminal licenses can be purchased and added to the host computer to increase the number of terminals that can be configured to communicate with the host computer. Communication by a terminal with the host computer can be carried by a network and does not violate item 1) above. [This Provision applies to the ITScriptNet® OMNI™ edition only.]
3. *Move* a registered license from one computer to another by unregistering the license from the licensed computer via the method provided in the software, then re-registering the license on a different computer. Compliance with paragraph 1 (Load and Use) above must be maintained. There are no restrictions as to the number of times a license can be registered and unregistered.

For Device-Based Licenses:

4. *Load and Use* the software on any number of computers without a serial number.
5. *License* each device that will be communicating with a computer. Each device license can be registered on a single terminal. Once registered on a terminal, a Device License can not be removed or assigned to a different terminal. A terminal with a Device License can communicate with any computer running the software whether that computer has a serial number or not.

For All Licenses:

6. *Copy* the software for back-up purposes only. You may make up to three (3) copies of the software for back-up purposes. All copies must contain the copyright notice printed on the label of the CD containing the original copy of the software.
7. *Transfer* the software and license permanently to another person if that person agrees to accept all of the terms and conditions of this Agreement. If you transfer the software, you must at the same time either transfer all copies of the software to the same person or destroy any copies not transferred.
8. *Terminate* this license by destroying the original and all copies of the software in whatever form.

YOU MAY NOT:

1. Loan, rent, lease, give, sublicense or otherwise transfer the software (or any copy), in whole or in part, to any other person, except as noted in paragraph 5 (Transfer) above.
2. Copy or translate the User Guide included with the software.
3. Copy, alter, translate, decompile, or reverse engineer the software, including but not limited to, modify the software to make it operate on non-compatible hardware.
4. Remove, alter or cause not to be displayed, any copyright notices or startup messages contained in the programs or documentation.

THIS LICENSE WILL TERMINATE AUTOMATICALLY if you fail to comply with the terms and conditions set forth above.

Term

The license is effective until terminated. You may terminate it at any time by destroying the programs together with all copies, modifications and merged portions in any form. It will also terminate upon conditions set forth elsewhere in this Agreement or if you fail to comply with any term or condition of this Agreement. You agree upon such termination to destroy the programs together with all copies, modifications and merged portions in any form.

Limited Warranty

What is covered?

Z-Space warrants to the original customer that (i) the CD on which the enclosed software is recorded is free from defects in materials and workmanship under normal use, and (ii) the software will perform substantially in accordance with the enclosed User Guide. EXCEPT AS SPECIFIED IN THIS PARAGRAPH, THERE ARE NO WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND THE PROGRAMS, DOCUMENTATION AND OTHER FILES ON THE CD ARE PROVIDED “AS IS.” (Some states do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.)

How long does this warranty last?

This Limited Warranty continues for sixty (60) days from the date of delivery of the software to the original customer (“Warranty Period”).

What will Z-Space do?

1. Z-Space will replace any CD which proves defective in materials or workmanship, if you return the CD postpaid to Z-Space during the Warranty Period with a dated proof of purchase.
2. Z-Space will, at its option, either replace the CD or correct any software that does not perform substantially in accordance with the enclosed User Guide if, during the Warranty Period, (i) you notify Z-Space in writing of any claimed defects in the software, (ii) you return the CD containing the software to Z-Space, and (iii) Z-Space is able to duplicate the defects on its computer system.
3. If Z-Space is unable to replace a defective CD or if Z-Space is unable to provide corrected software within a reasonable time, Z-Space will, at its option, either replace the software with functionally equivalent software or refund the license fees paid by you. THESE ARE YOUR SOLE AND EXCLUSIVE REMEDIES for any and all claims that you may have against Z-Space arising out of or in connection with this product, whether made or suffered by you or another person and whether based in contract or tort.
4. IN NO EVENT WILL Z-SPACE BE LIABLE TO YOU OR ANY OTHER PARTY FOR DIRECT, INDIRECT, GENERAL, SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY OR OTHER DAMAGES ARISING FROM THE USE OF OR INABILITY TO USE THE SOFTWARE OR FROM ANY BREACH OF

THIS WARRANTY, EVEN IF Z-SPACE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. (Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above exclusion or limitation may not apply to you.) In no event shall Z-Space's total liability exceed the amount you paid in license fees for the right to use a single copy of this software. Z-Space's software pricing reflects the allocation of risk and limitations on liability contained in this Limited Warranty.

What additional provisions should I be aware of?

1. Because it is impossible for Z-Space to know the purposes for which you acquired this software or the uses to which you will put this software, you assume full responsibility for the selection of the software, and for its installation and use and the results of that use.
2. While every reasonable effort has been made to insure that you will receive software that you can use and enjoy, Z-Space does not warrant that the functions of the software will meet your requirements or that the operation of the software will be uninterrupted or error free. Due to the complex nature of computer programs, the programs in this package (like all large programs) will probably never be completely error-free.
3. This Limited Warranty does not cover any CD which has been the subject of abuse or damages, nor does it cover any software which has been altered or changed by anyone other than Z-Space.
4. Z-Space is not responsible for problems caused by changes in the operating characteristics of the hardware or operating system software you are using which are made after the release date of this version of ITScriptNet® with any other software.
5. You agree to comply with all applicable international and national laws that apply to the products as well as end-user, end-use and destination restrictions issued by governments.
6. If the SOFTWARE is labeled as an upgrade, you must be properly licensed to use a product identified by Z-Space as being eligible for the upgrade in order to use the software. Software labeled as an upgrade replaces and will disable the original software which was initially loaded on the computer. After upgrading, you may no longer use the software that formed the basis for your upgrade eligibility. You may use the resulting upgraded product only in accordance with the terms of this license agreement and only with a computer that has also registered the original software.
7. This agreement constitutes the entire agreement between you and Z-Space and supercedes any prior understandings and agreements, either oral or written. It shall be interpreted under the laws of the State of Ohio.
8. This warranty gives you specific rights and you may also have other rights which vary from state to state.
9. No action for breach of warranty may be commenced more than one (1) year following the expiration date of the above Limited Warranty.

Should you have any questions concerning this Agreement, you may contact Z-Space by writing to Z-Space Technologies, Inc. 26933 Westwood Road, Suite 100, Westlake, Ohio 44145.

Introduction

Welcome

Welcome to **IT*Script*Net**®, the easy-to-use software that allows you to quickly and easily create data collection solutions for portable terminals. **IT*Script*Net** is designed to be easy-to-use, yet powerful enough to support the most sophisticated applications. With **IT*Script*Net** you will be designing data collection programs in no time! Designing your program is simple--you can see the flow of the data collection program graphically and can dictate exactly how each of your responses and fields are to be answered. **IT*Script*Net** includes a New Program Wizard that will immediately create data collection programs for common applications or will serve as a starting point for customized solutions. A group of Ready-To-Go® Applications is provided to let you start collecting data right away. **IT*Script*Net** even includes a Simulator so you can try out your program at the PC before deploying your solution to the actual data collection terminals.

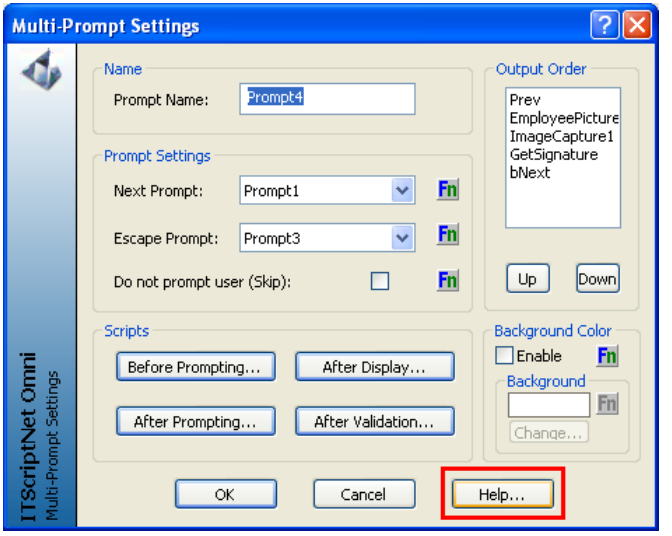
IT*Script*Net supports a variety of portable data terminals. For a complete listing of supported terminals, please refer to our web site **http://www.z-space.com.**

There are 3 editions in the **IT*Script*Net** product family. **IT*Script*Net Batch** is for creating Batch data collection programs where each piece of data collected corresponds to a prompt in the program. The **IT*Script*Net Batch Plus** product takes Batch Data Collection to the next level by adding Multi-Prompts and a rich set of screen elements, printing support, and other developer features. **IT*Script*Net® OMNI** is the RF-enabled evolution of the **IT*Script*Net** product family that includes wireless communication functionality for radio frequency (RF) applications, and allows you to create real-time RF data collection solutions. Every feature in **IT*Script*Net® Batch** and **IT*Script*Net® Batch Plus** is included in **IT*Script*Net® OMNI**, making it the all-inclusive tool for data collection.

Regardless of the features you need for your data collection solution and the **IT*Script*Net** product you are using, you will find that **IT*Script*Net** is easy to use and extremely powerful and flexible.

Online Help

IT*Script*Net provides online Help. All of the topics discussed in the User Guide can be accessed through the online help. There are two types of online help. The HTML Help is built-in to the program and can be accessed from the main menu and from any **Help…** button throughout the program. The second type of help is the context-sensitive help designed to quickly provide basic information on any screen control by pressing F1 or by clicking on the **'?'** button and then clicking on a screen control.



Technical Support

If you need technical support on this product, please contact your reseller or hardware manufacturer. You can also access technical support at **http://www.z-space.com**, or by emailing **support@z-space.com**, or by calling (440) 899-7370 during normal business hours.

ITScriptNet Licenses

IT*Script*Net can be licensed in two ways: by PC, or by Device.

For PC-Based licensing, **IT*Script*Net** requires a license for each PC it is installed on in order to run without a demo mode limitation. By purchasing and registering the proper **Batch** product licenses the user will have full access to all of the functions and features of the licensed software. Each license can only be registered on one computer. The user may move licenses from one computer to another by un-registering the license and re-registering as many times as necessary.

For Device-Based licensing, **IT*Script*Net** requires a license for each device in order to run without a demo mode limitation. A terminal with a device license can communicate with any computer running the software whether that computer has a PC license or not. Device licenses, once registered on a terminal, can not be removed or moved to a different terminal.

IT*Script*Net Batch Plus programs running on properly equipped terminals**,** through the use of certain Print functions, can send data to be printed to certain wireless printers. Print methods include IrDA, 802.11b and Bluetooth. No additional licensing is required for the terminals or printers.

Demo Mode Limitations

Programs designed using the demo version of **IT*Script*Net** are limited to collecting and downloading 10 records. This is typically enough to test the program design and communication capabilities of the software.

PC Licenses

IT*Script*Net PC Licenses consist of two (2) levels of licensed software products:

Program Designer

The Program Designer is used to develop data collection programs for the terminals and Pocket PC and Windows CE devices. The software is always licensed on a per PC basis. A user may develop unlimited programs with a single Designer license. The Program Designer license also includes the Upload and Download communication utilities, which allow the user to upload data collection programs and related files to terminals and to download collected data from the terminals to the PC. There is no limit on the number of terminals that can communicate with a Program Designer licensed PC.

Runtime

An **IT*Script*Net Runtime** license can be obtained for a PC that is only required to upload data collection programs and related files to terminals, and to download collected data from the terminals to the PC, and does not need program design features. There is no limit on the number of terminals that can communicate with a Runtime licensed PC. The Runtime Serial Number is registered from the Upload Utility.

Device Licenses

IT*Script*Net Device Licenses apply to portable data collection terminals, and allow the terminal to download collected data to any PC running the Upload or Download utilities without requiring a PC license on that computer. Please note that you still must design your data collection program with a Program Designer that has a PC license. Otherwise your program will still have the demo mode limitation.

Installing the Software

Installation Requirements

IT*Script*Net has the following installation requirements:

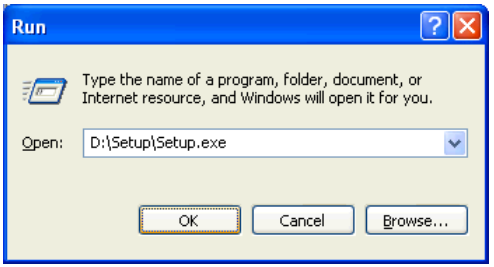
- Windows 98, Windows Me, Windows NT4SP4, Windows 2000, Windows XP, or higher
- Minimum RAM: 16 MB (32 MB for Windows NT, Windows 2000, Windows XP)
- Minimum hard drive space required: 30 MB (60 MB for full installation)
- Screen resolution of 800x600 or higher recommended.
- One or more supported portable data collection terminals configured for batch data collection or RF data collection
- Available serial or USB Port for communication with the portable data collection unit for batch data collection
- Wireless LAN/Access Point configured for communication with portable terminals for RF data collection.
- The online help system requires Internet Explorer 4.0 or higher for HTML. If you do not have Internet Explorer 4.0 or higher, the online help system may not function, although the rest of the program will.

If your system does not meet these requirements, the program may not run properly.

Installing ITScriptNet

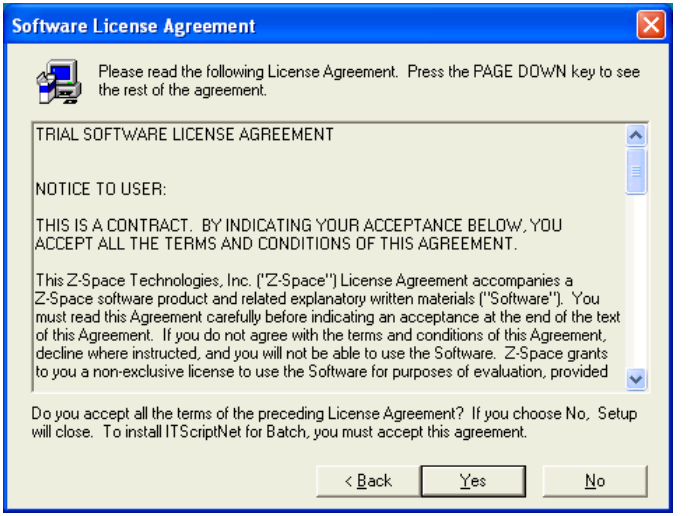
IT*Script*Net uses an installation setup program to step you through the installation process.

To install the program, put the CD-ROM in your drive. The setup program will run automatically. If it does not, then select **Start->Run**, and enter **x:\setup\setup.exe** where **x:** is the letter of your CD-ROM drive.

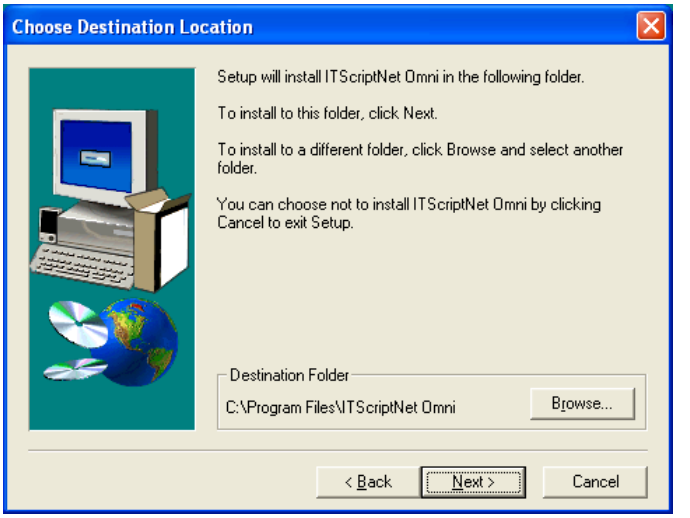


After the setup program is done with its initialization, you will see the *Welcome* screen. Press **NEXT** to continue the installation.

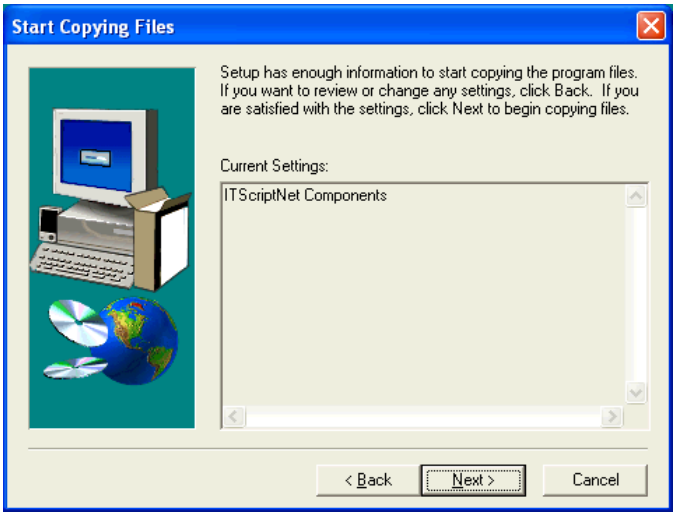
The next screen is the *Software License Agreement* screen. Please read the license agreement, and if you agree, press **Yes**. If you choose No, the installation program will exit.



The next screen shows the location where the program will be installed. You can also **Browse** to put the program into a different location. Click **Next** to accept the destination folder.



When the setup program has enough information to start copying files, you will see a list of the components to install. This list may vary somewhat depending on your PC's configuration, but will always install the **IT*Script*Net** components.



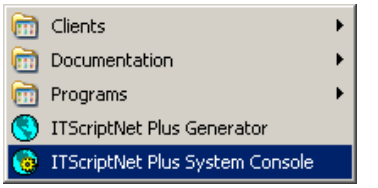
Click **Next** again to continue with the installation process. Depending on your operating system and system configuration, the installation program may alert to reboot your PC and to rerun the installation program. This will be required if the DCOM Microsoft component needs to be installed (DCOM is included in the operating system for Windows 98 and higher).

Congratulations! Setup is complete!

After the program has been successfully installed, you can optionally launch the **IT*Script*Net** System Console at the end of the setup process, or run **IT*Script*Net** from the Start button. The **IT*Script*Net** System Console is the central location for launching all **IT*Script*Net** programs. There are communications utilities for sending programs and data to the terminals and receiving collected data in batch mode, installing clients on the portable devices, licensing and more. You can find more information about working with a specific model of data collection terminal in the device-specific User Guides.

System Console

The **IT*Script*Net** installation process will place a shortcut to the **IT*Script*Net** System Console on your desktop. Alternately, you can navigate (as pictured at the right) from the Start button to: Start→Programs→ITScriptNet Plus → Plus System Console.



Console Overview

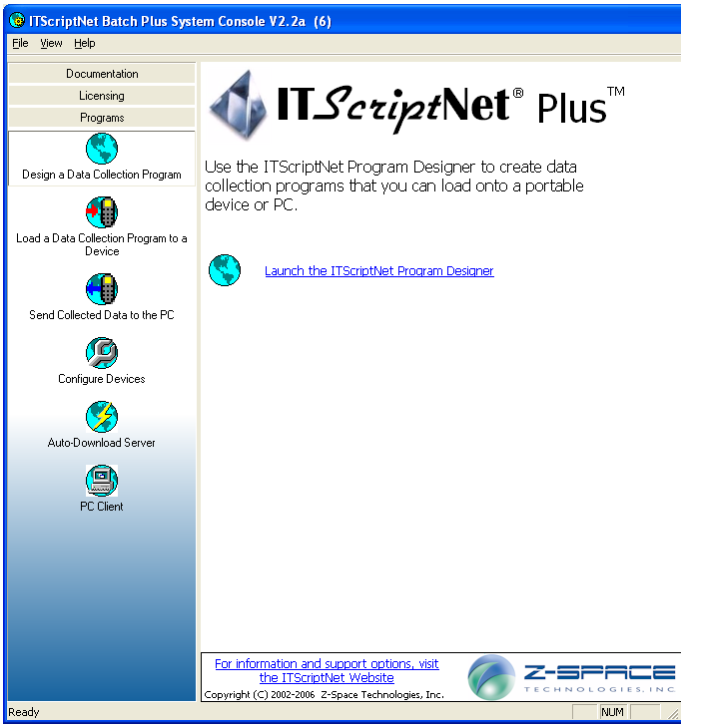
The *System Console* allows you to easily view and navigate the setup, program design, and communication links in **IT*Script*Net**.

The initial view (shown at right) displays a list of the Program components which make up the **Plus** software. Selecting an item in the pane on the left will display instructions and a link to that component in the pane on the right.

Each of the four buttons at the top of the left pane will drop down more component links for Documentation and Licensing.

Each of these component links may also be accessed by navigating to Start→ Programs→ITScriptNet Plus.

For detailed information on each component in this view, please refer to the corresponding sections in this User Guide.



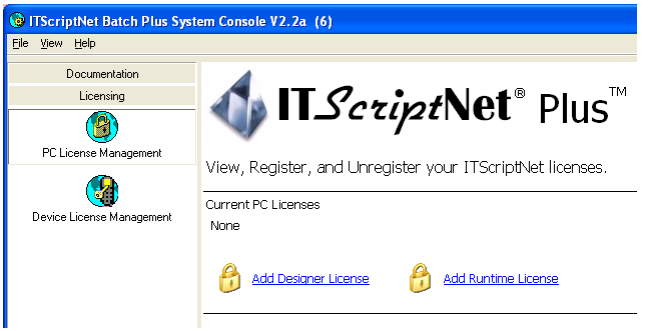
If you have downloaded a demo version of the software and prefer to run the **Plus** Designer before registration of the product, a registration window will appear. You will be able to run the software prior to registration by clicking on the **Continue** button to skip the registration process and begin the program.

What To Do First

After **IT*Script*Net** has been installed to your computer, you must register the software in order to remove the demo mode limitation:

Licensing

In the *Console* view, click on the **Licensing** button and the Registration window will appear (shown at right).



There are three license registrations for the **IT*Script*Net** product:

- Designer program
- Runtime communication utility
- Terminal Packs

In the Licensing view, click on each link next to the yellow padlock icon to register your **IT*Script*Net** products.

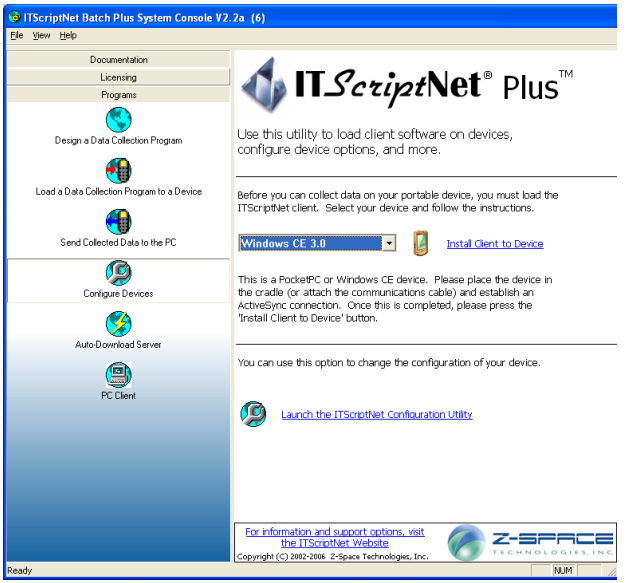
For more detailed instructions on the Registration process, please refer to the **Registration** section in this User Guide.

Device Configuration

After you have registered your products, your will need to configure your devices. The *System Console* provides a link to make this process quick and easy!

In the left pane of the *Console*, select **Configure Devices**. In the right pane, a drop-down box is available which displays a list of device clients installed with your **IT*Script*Net** software. Select your device type from the list.

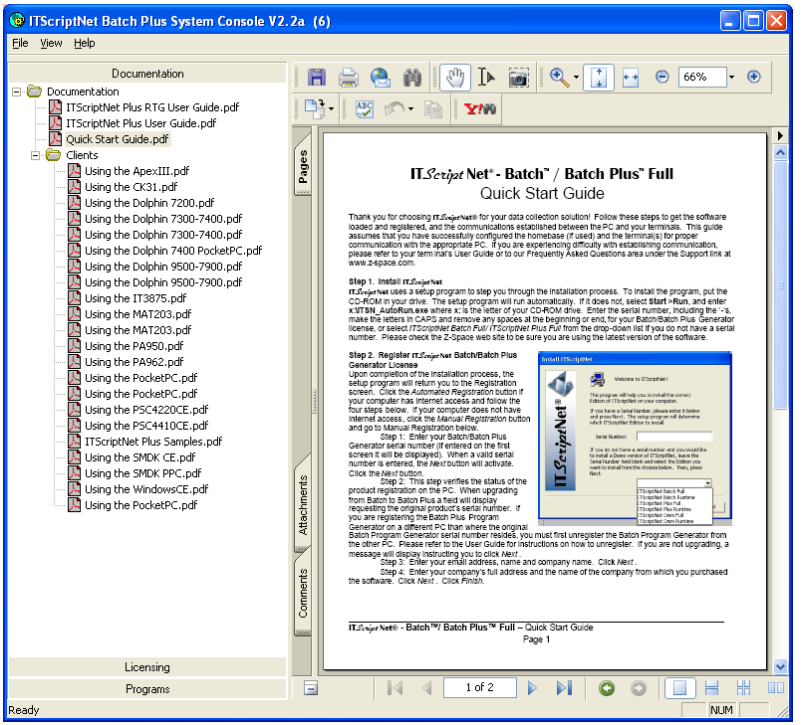
Adjacent to the device drop-down box is a link to **Install Client to Device**. Clicking this link will launch the client installation. Follow the on-screen instructions to install the client to your device.



Documentation

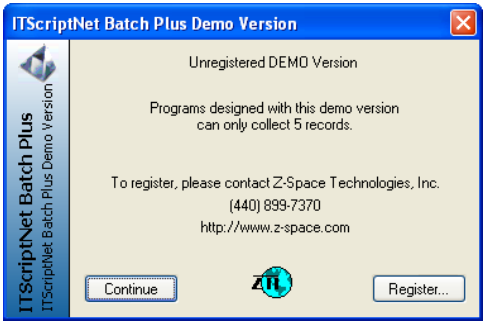
The *System Console* **Documentation** link displays a list of the PDF documents installed with the software. Selecting a document link in the left pane displays the document in the right pane. This is a fast and easy way to access the many **IT*Script*Net** guides available.

These documents can also be accessed from:
Start→Programs→ ITScriptNet Plus→ Documentation.

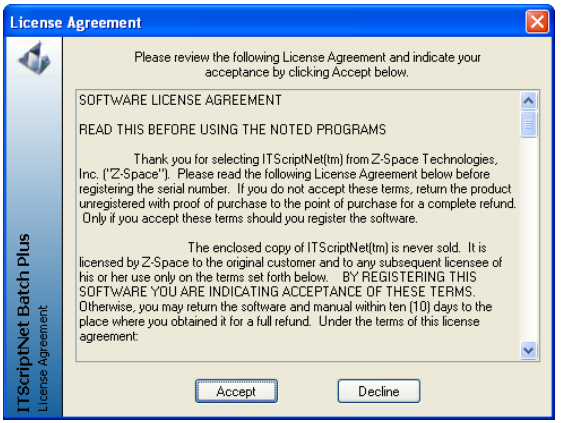


PC License Registration

If you are using PC Licenses to register **IT*Script*Net**, you can also access registration from the System Console. You will need to register the software in order to remove the communication limitations from demo mode. If you have downloaded a demo version of the software, you will be able to run the software prior to registering by clicking on the **Continue** button to skip the registration process and begin the program. If you do not register, the software will remain in demo mode.



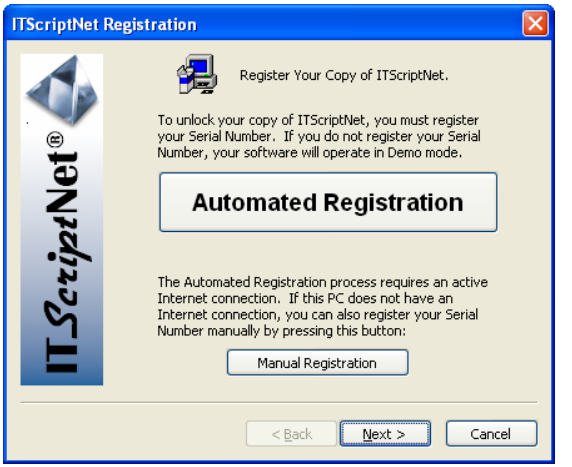
Registration need only be done once and takes only a few minutes. It's as easy as 1-2-3! Click the **Register** button to go to the *License Agreement* screen and read the license agreement. By registering the software you are licensed to full use of the software. If you accept the terms of the license agreement, click the **Accept** button to continue with the registration. If you click the **Decline** button, you can continue to use the software under the terms of the trial license agreement that you accepted when installing the software.



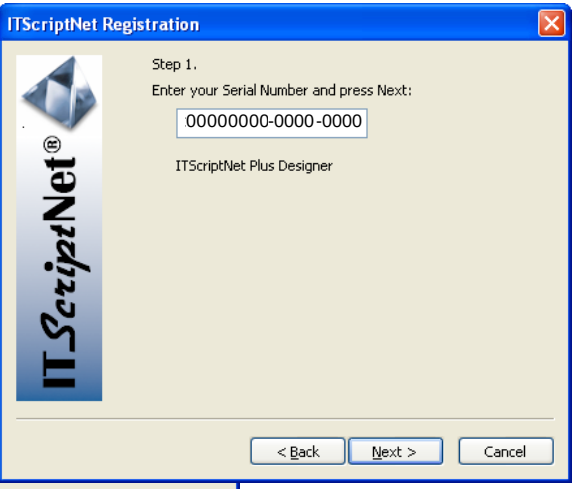
Automated Registration

To register your copy of **IT*Script*Net**, click on the Automated Registration button. This registration process requires an active Internet connection.

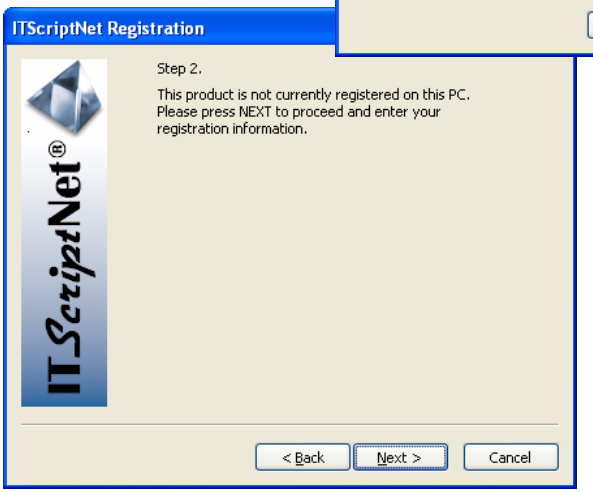
If the software is installed on a computer that does not have an Internet connection, please skip to the Manual Registration section which follows.



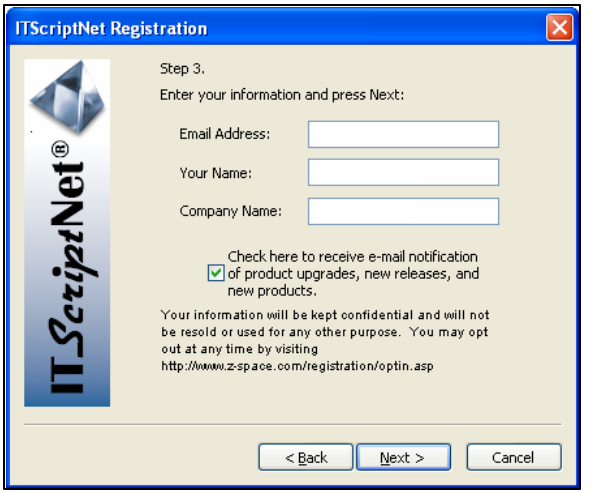
Step 1 requires that you enter the serial number that you received via email or with your boxed product. Enter the serial number *exactly* as it appears including the dashes. When you have entered the serial number correctly, the **Next** button will activate.



Step 2, please read the message, then click on **Next.**



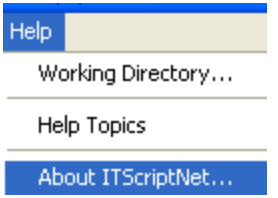
Step 3, fill in the required information. Click **Next.**



The next window to appear will display a message indicating that your software registration is complete.

If you have created any data collection programs in an un-registered demo mode, you will need to re-save the programs using your registered software. You will also need to send these programs to the terminals which will then allow the terminals to collect an unlimited number of records.

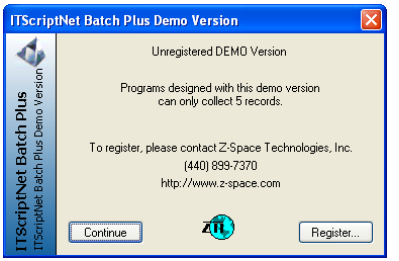
If you ever need to know your serial number, you can find it in the *About* dialog box accessible from the *Help* menu.



Manual Registration

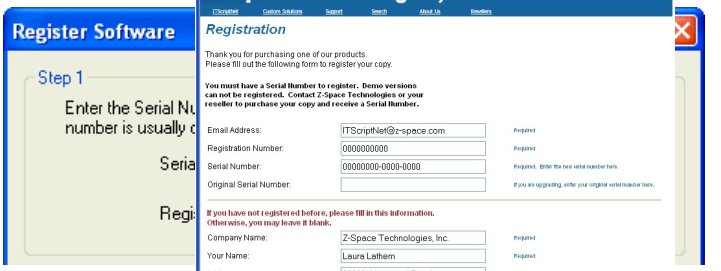
If the software is installed on a computer that does not have an Internet connection, you can register manually from a different computer which does have Internet access, or call Z-Space Technologies, Inc.

Manual Registration need only be done once and takes only a few minutes. It's as easy as 1-2-3! To begin Manual Registration, click the **Register** button, accessed through the Help/About menu item, to go to the *License Agreement* screen and read the license agreement. By registering the software, you are licensed to full use of the software. If you accept the terms of the license agreement, click the **Accept** button to continue with the registration. If you click the **Decline** button, you can continue to use the software under the terms of the trial license agreement that you accepted when installing the software.



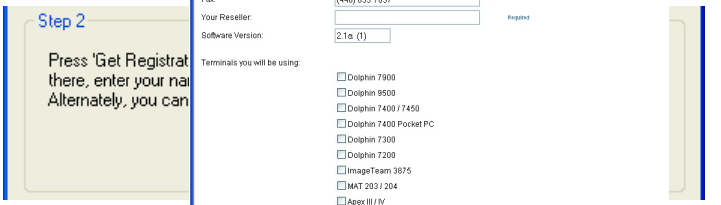
Click on the Manual Registration button, as shown at right, to begin the manual registration process.

The *Register Software* screen shows the 3 steps needed to register manually.



Step 1 requires that you enter your serial number. The serial number is normally found on the bottom of the box, and on the sleeve of the CD-ROM.

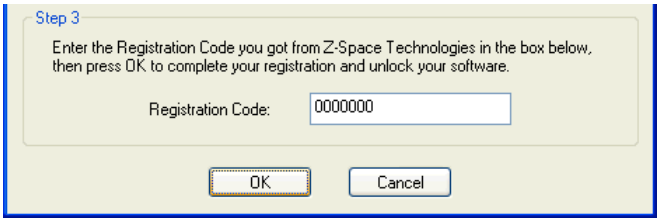
Enter the serial number *exactly* as it appears, including the dashes. When you have entered the serial number correctly, the **Get Registration Code Online…** button will activate.



Step 2, click on the **Get Registration Code Online…** button to access the Z-Space registration web site automatically if your PC has Internet access, or, if you need to use a different PC for web access, go to: **http://www.z-space.com/registration** to manually enter the serial number and registration number into the web site registration form.

Fill out the online registration form (shown at right) and click the **Submit** button. In the next view, not shown here, the web site will display the information you have entered and ask you to confirm that everything is correct. After you click the **Confirm** button on the web site, the registration web site will generate and display the **Registration Code** to unlock your software.

Step 3. On the computer on which the software is installed, type (or paste) the registration code generated by the registration web site into the input box in Step 3 (below). Make sure you have entered the Registration Code number exactly as displayed on the registration web site. Press the **OK** button to complete the registration process.

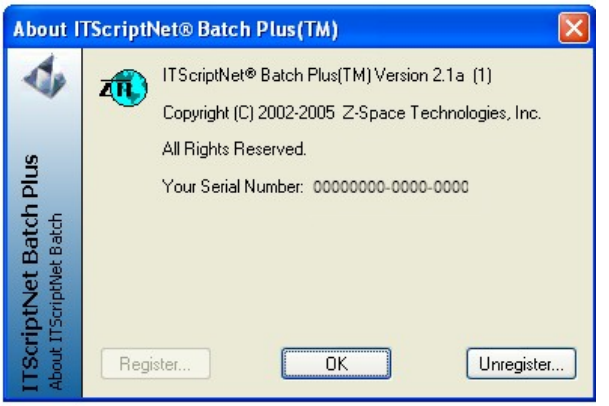


You can also call Z-Space Technologies at (440) 899-7370 and our friendly and helpful staff will assist you.

Tip: You can select the Registration Number on the website and right-click your mouse to bring up a menu so that you can click the copy menu item. Then, back in **IT*Script*Net**, you can right-click with your mouse on the Registration Number field and select Paste. This will allow you to avoid having to actually type this number.

Unregistration

Because the registration for **IT*Script*Net** is specific to each PC, you will need to un-register the software in the event that you need to move the software's registration to a different PC. The un-register process is accessible from the *About* box and is essentially the reverse of registration. The *About* screen displays your serial number and the selection button to Unregister your software.



Click on the **Unregister** button.

Automated Unregistration

To unregister your copy of **IT*Script*Net,** click on the Automated Unregistration button which will take you to the Unregister page on the Z-Space web site. The unregistration process requires an active Internet connection.

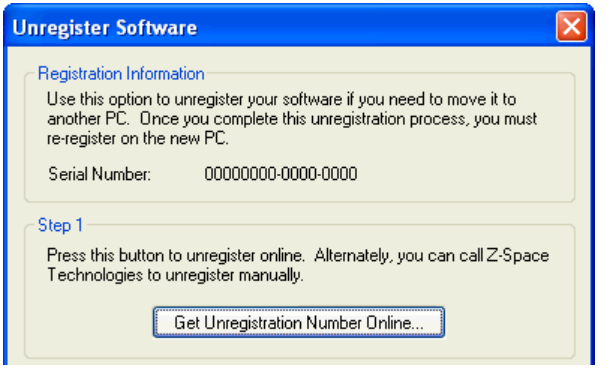
The software will ask you to confirm that you want to unregister your software. Click OK in the next message box, and your serial number is released and can be used to register the software on another PC.

If the software is installed on a computer that does not have an Internet connection, please click on the Manual Unregistration button on the bottom, as shown at right. Follow the procedure, which is similar to the registration process by manually entering the serial number at:
http://www.z-space.com/registration/unregister.asp
or call Z-Space Technologies at (440) 899-7370 for assistance.

Manual Unregistration

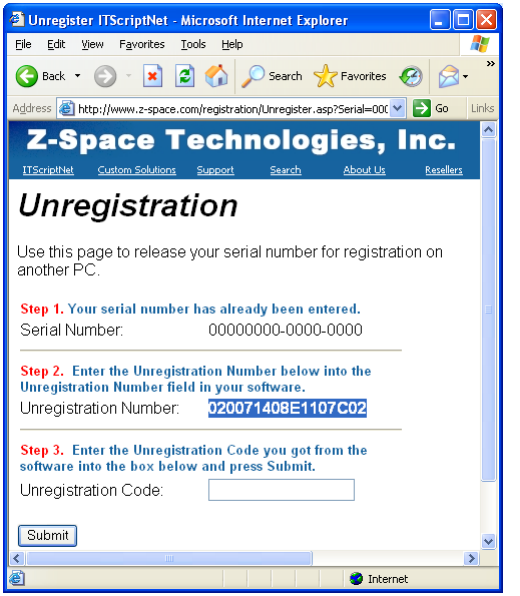
If the software is installed on a computer that does not have an Internet connection, you can unregister manually from a different computer which does have Internet access.

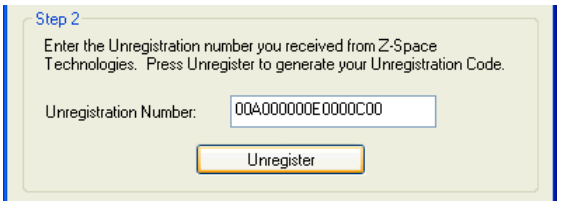
Step 1, click on the **Get Unregistration Number Online…** button.



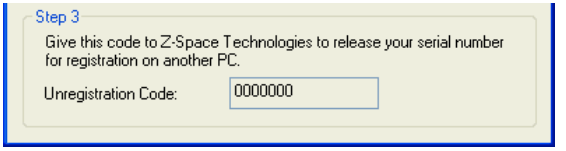
This will take you to the Unregister page on the Z-Space web site.

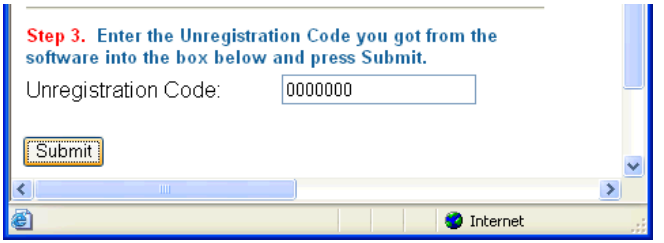
Step 2, copy the Unregistration Number from the web site shown for Step 2 into the Step 2 Unregistration Number field in ITScriptNet. Tip: You can select the Unregistration Number on the website and right-click your mouse to bring up a menu so that you can click the copy menu item. Then back in **IT*Script*Net**, you can right-click with your mouse on the Unregistration Number field and select Paste. This will allow you to avoid having to actually type this number. When the Unregistration Number has been entered, click on the **Unregister** button. This will un-register your copy of **IT*Script*Net** on your PC, but Step 3 is also necessary to complete the process.





Step 3, copy the Unregistration code from the Step 3 box in **IT*Script*Net** to the website's field for Step 3 and press the **Submit** button on the web site.





The web site will confirm that your software has been un-registered. Your serial number is now released and can be used to register the software on another PC.

As with registration, you can manually enter the serial number at **http://www.z-space.com/registration/unregister.asp** or call Z-Space Technologies at (440) 899-7370 for assistance.

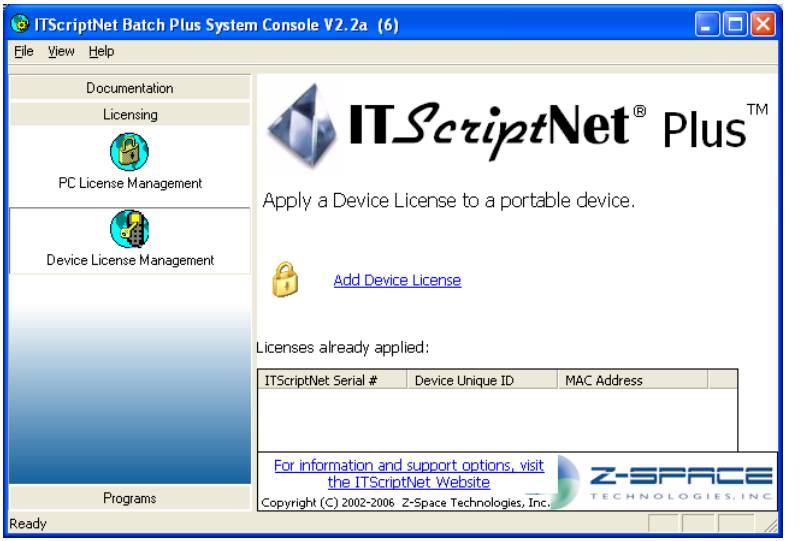
Device License Registration

If you are using Device Licenses instead of PC Licenses, you must apply the licenses to each terminal using the **IT*Script*Net** System Console. All Device License serial numbers begin with a T.

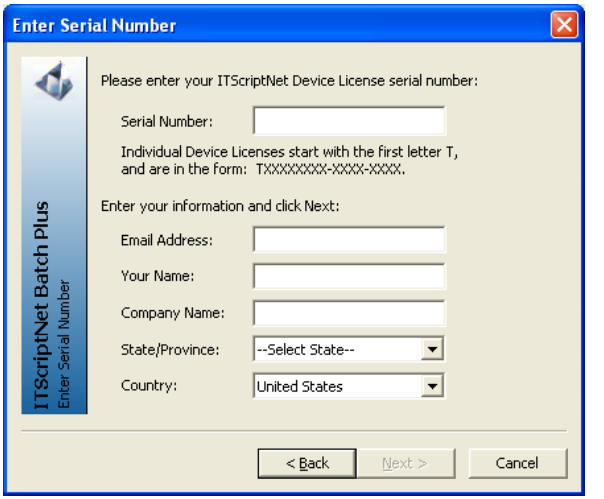
To begin, launch the ITScriptNet System Console and click the **Licensing** Tab, followed by the **Device License Management** button.

Click **Add Device License** to start the licensing process.

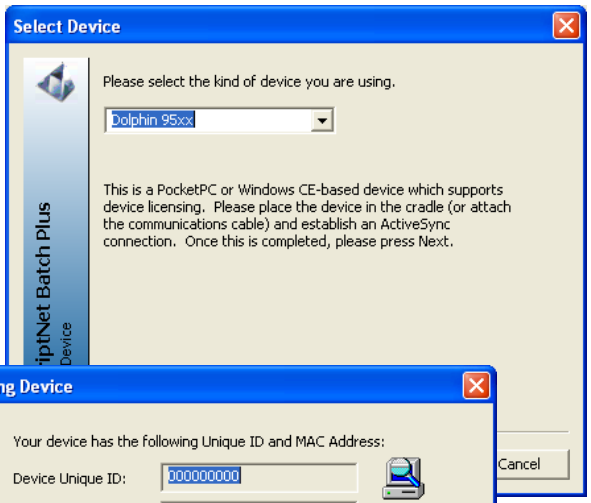
You will be prompted to read and accept the license agreement, then you will be presented with some instructions. Please read these and then click Next.



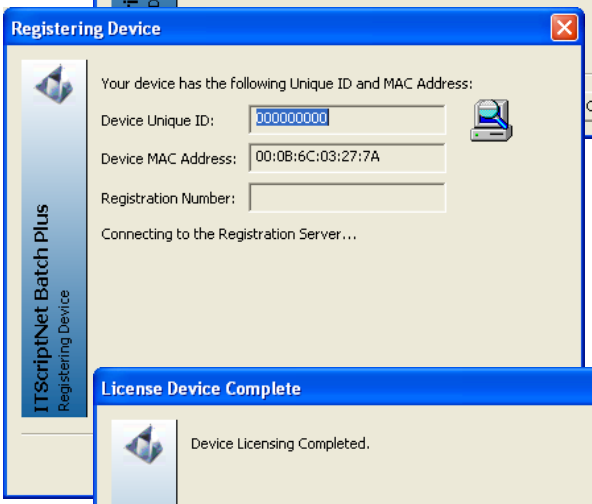
Enter your Device License Serial Number and Contact information, and press Next.



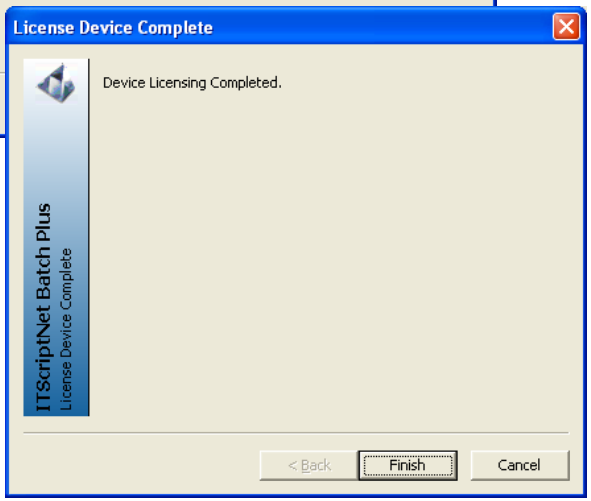
Select the type of device you are using. Please read the instructions below the device type. For PocketPC and Windows CE-based devices, you will need to create an ActiveSync connection to your device and install the client first. You do not need to create a partnership, but simply a Guest connection is OK.



The registration process will attempt to connect to the device and retrieve its Unique ID and MAC Address for registration.



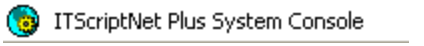
If the registration is successful, you will see the completion message. If there is a problem, you will have the option of retrying your registration.



Installed Components

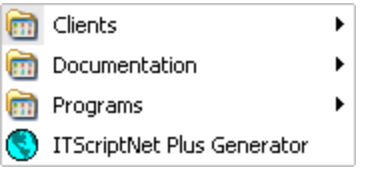
After the program has been successfully installed, you can run **IT*Script*Net** from the Start button. You will notice several folders and programs in the **IT*Script*Net** program group.

System Console



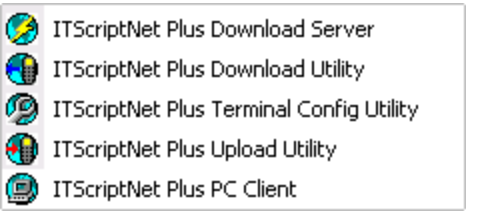
The **IT*Script*Net** System Console is the starting point for all of the other **IT*Script*Net** programs and utilities. From here you can design programs, install clients, manage licenses, and upload and download data.

Program Generator



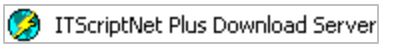
The **IT*Script*Net Plus** Generator program is installed in the root of the program group. The Generator is the application used to design the data collection programs. The program group also contains folders for Clients, Documentation, and Programs.

Programs



In addition to the Generator, the **IT*Script*Net Plus** package includes several other applications. These are accessible from the Program folder in the program group. Each application that is part of the **IT*Script*Net** package has a dedicated section describing it in detail later in this User Guide.

Download Server

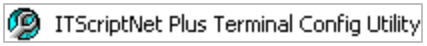


The Download Server is the application that supports **IT*Script*Net's** Autodownload feature. This application, when running, can receive and process data from a terminal simply by placing the terminal in its Home Base, cradle, or connecting it to its communication/charge cable.

Download Utility

The Download Utility allows data to be received from the terminal and processed.

Terminal Configuration Utility



This utility application allows the configuration for a terminal to be controlled from the PC. Most of the configuration settings for a terminal can also be controlled from the Client application's Configuration screen.

Upload Utility

The Upload Utility sends a program to the terminal along with any validation files or support files that go with the data collection program.

PC Client

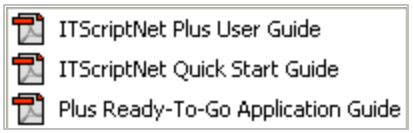
The PC Client utility allows an operator to collect data on a PC using the same data collection programs as the portable terminals.

Clients

The Client is the **IT*Script*Net** component that runs on the terminal and receives data collection programs designed with **IT*Script*Net**, runs the data collection programs to collect data, and then sends the data back to the PC for processing. All the Windows CE and PocketPC devices require a Microsoft ActiveSync connection to be established prior to installing the client application on the terminal. Once established, simply run the Client Install program for your terminal type. Documentation for each specific device is located in the **Documentation -> Client Guides** program group.

Documents

The documentation for **IT*Script*Net** is always supplied electronically and installed as part of installation. You can access **IT*Script*Net's** documentation from the Documentation folder in the program group.



Getting Started

This chapter provides a brief overview of **IT*Script*Net** and emphasizes the use of the Ready-To-Go® Applications and the New Program Wizard to get started creating data collection solutions immediately!

Data Collection Solution Overview

Here is a list of the basic steps to create and deploy a data collection solution with **IT*Script*Net**. Each of the basic steps includes references to the areas in this User Guide with more information.

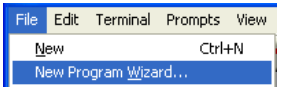
	Step	See this Section for More Info
1	Install ITScriptNet	*Installing ITScriptNet*
2	Register	*Installing ITScriptNet - Registration*
3	Design Data Collection Program – you can use a Ready-To-Go Application, the New Program Wizard, or the sample programs to get started	*Getting Started*
4	Test Data Collection Program in Simulator	*Terminal Menu - Simulate*
5	Install the ITScriptNet Client on your Terminal	See the Section specific to your Terminal
6	Upload Program (Send to Terminal) and Collect data using your data collection program	*Terminal Menu – Send Program to Terminal* OR *Upload Utility*
7	Download Data (Receive data from Terminal)	*Terminal Menu – Receive a File from the Terminal* OR Down*load Utility*

Ready-To-Go Applications

The Ready-To-Go Applications are a set of data collection programs designed with **IT*Script*Net**. Several may have come pre-loaded on your terminal! Ready-To-Go Applications are never limited to 10 records so they are perfect for demos and to use right away to get started collecting data. The Ready-To-Go Applications are installed with **IT*Script*Net** and are located in the Ready-To-Go-Apps subdirectory where you installed **IT*Script*Net**. The Ready-To-Go Applications might also be a good starting point for the design of your data collection program. However, if you do modify any of the Ready-To-Go Applications, you will need to purchase and register **IT*Script*Net** in order to remove the demo communication limitations. Details about the Ready-To-Go Applications can be found in the Ready-To-Go Application Guide that is installed with **IT*Script*Net**.

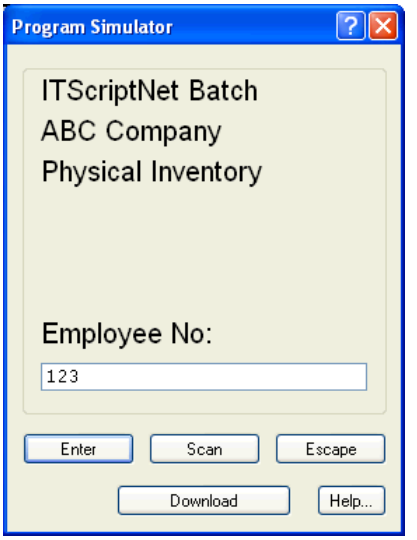
New Program Wizard

The New Program Wizard is designed to get your programs up and running quickly. In as few as 8 clicks, you can have a basic data collection program up and running. Select *New Program Wizard* from the *File* Menu. Follow the instructions on the screen as the Wizard guides



you through the process of creating a data collection program.

Simulator



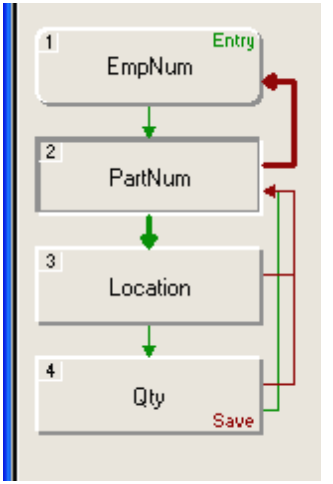
After creating a data collection program, it is a good idea to use the simulator. This will allow you to review the prompts so that you can easily identify any changes you wish to make. It is easier and faster to test your program from the simulator screen than it is to test with the portable terminal. You can start the simulator by clicking on the *Simulate* menu item found under the *Terminal* main menu item, or by clicking on the simulate icon on the toolbar.

Program Design Concepts

This section describes the basic concepts of designing data collection programs with **IT*Script*Net**.

Prompts

The data collection programs that **IT*Script*Net** creates are based on Prompts. Each program contains a series of one or more prompts. The program flows from one prompt to the other as the user enters data. One type of prompt is called a *Single-Prompt.* Each prompt may require input from the user or be used to display information. All three editions of **IT*Script*Net (Batch, Plus,** and **OMNI)** support Single-Prompts.



There is also a second type of prompt called Multi-Prompt. **IT*Script*Net Plus** and **OMNI** support the use of Multi-Prompts. A Multi-Prompt can utilize a richer set of design elements. There are two types of elements, input and display, that enhance the user's efficiency by making it easier to navigate the program and by entering multiple pieces of data on a single screen.

Both Single-Prompts and Multi-Prompts are explained in full detail in their own sections of this User Guide.

Looping

Generally, prompts for data collection are presented to the user one after another in a logical order. The order of the prompts within the program determines the order of the prompts presented to the user at data collection time. When the user gets to the end of the prompts, the data collected is

saved and the user will need to start again at the first prompt to collect another set of data. Each pass through the prompts generates a record in the stored data. When downloaded to a text file, each record will be represented by one row of text. When downloaded to a spreadsheet or database, each record of collected data corresponds to a row in Excel or a record in a database.

In the example shown, there are four prompts: EmpNum, PartNum, Location, and Qty. The short green arrows show the flow of the program. In this case, the user will be prompted for PartNum after successfully entering the EmpNum. After PartNum comes Location, then Qty.

However, when the user successfully enters the quantity on the Qty prompt, the program loops back up to PartNum and not EmpNum. The green arrow from the Qty prompt up to the PartNum prompt indicates that the looping structure for this example program skips the EmpNum after the first loop through the prompts. **IT*Script*Net** allows you to design programs to collect data *efficiently*. If an employee enters his employee number, EmpNum, once, it is unlikely that it will be necessary for him to enter it for every record. The data collection process can be made more efficient by minimizing entry of repetitive data. In this example, the designer of the program has decided that after the first time through the loop, the program should not prompt for EmpNum. Since the user is not prompted for EmpNum, the program holds the response and saves it as if it were entered for all subsequent prompting loops.

The green arrows show the prompt looping structure as the program flows from one prompt to another on a successful prompt response.

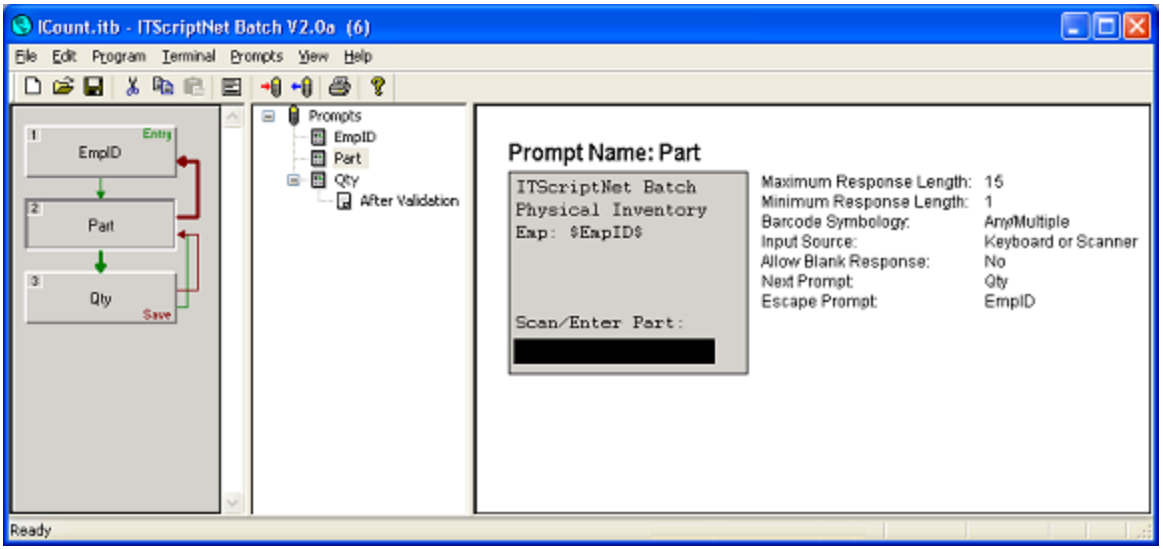
The red arrows indicate the program flow when the user presses the Escape key to exit the prompt. The Escape will abort the current prompt and jump to the prompt shown by the red arrow. In our example, if the user escapes during the Location prompt or during the Qty prompt, the program will move back to PartNum. The user can then respond to the PartNum prompt (the EmpNum will still be retained). If the user escapes while on the PartNum prompt, however, the program will move back to the EmpNum, or the beginning of the program. If the user escapes from the first prompt, the program will exit the data collection program entirely and return the user to the main terminal menu.

The program flow, or looping, can be more complex than described here. Using In-Prompt Scripts (described later in this User Guide), the data collection program can be made to have conditional branching. With conditional branching, the state of the program, the data collected, and/or the value of program variables determine the next prompt (or escape prompt) while the program is running. The green and red arrows that indicate program flow do not reflect possible conditional branching defined in the prompt's scripts.

Single-Prompt

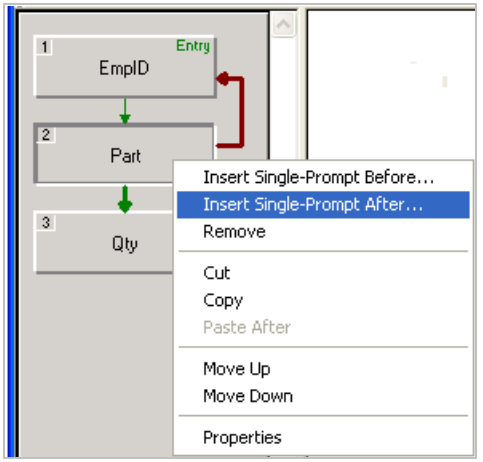
Design Environment

A Single-Prompt is a prompt in **IT*Script*Net** that collects one piece of data. **IT*Script*Net Batch Plus** and **IT*Script*Net OMNI** also have a Multi-Prompt feature that allows for collecting more than one piece of data per prompt. The **IT*Script*Net** design environment contains several sections each geared towards helping you design data collections programs.



Prompt Flow

The left side of the main window is the prompt list. The prompt list is like a flow chart showing the order of the prompts. Each prompt is listed. There is a green arrow connecting the prompts showing the path from prompt to prompt after a successful prompt response. Red arrows show the path the prompt specified as the Escape Prompt. The Prompt area allows you to control several prompt-level actions.



Creating a Single-Prompt

To create a new prompt, right-click on a prompt and select the Insert Single-Prompt After or Insert Single-Prompt Before menu item from the popup menu. You

can also select one of these Insert menu items from the Prompts main menu.

Copy a Single-Prompt

You can also create a new prompt by right-clicking on a prompt, selecting copy, right-clicking on the same or another prompt and selecting Paste After. You will be prompted to enter a new name for the new prompt. Copying prompts can save time since all of the settings and scripts for the prompt will be copied to the new prompt.

Remove a Prompt

Prompts can be deleted by right-clicking on a prompt and selecting the Remove menu item. You will be asked to confirm deleting the prompt since removing a prompt can not be undone. The Remove menu item is also available from the Prompts main menu.

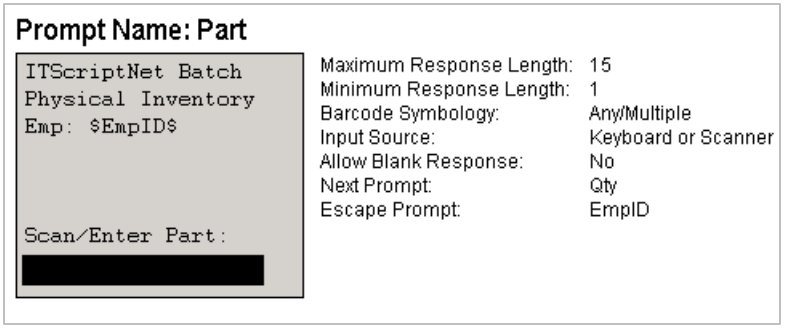
Move Up/Move Down

You can move prompts relative to each other with the Move Up and Move Down menu items. Access the Move Up and Move Down menu items from either the right-click menu or from the Prompts main menu item.

Properties

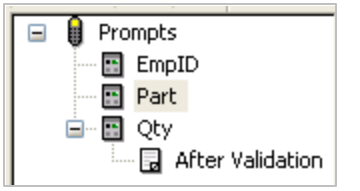
The Properties menu item will bring up the Prompt Settings screen. This screen defines the properties, or settings, for the prompt. Each of these settings is described in a later section.

Main Design Area



For Single-Prompts, the main design area provides a visual representation of the prompt. The prompt's settings are listed next to the visual representation for quick reference. The representation of the prompt will be slightly different depending on the terminal type. You can double-click on the prompt shown to bring up the Prompt Settings screen.

Script Tree



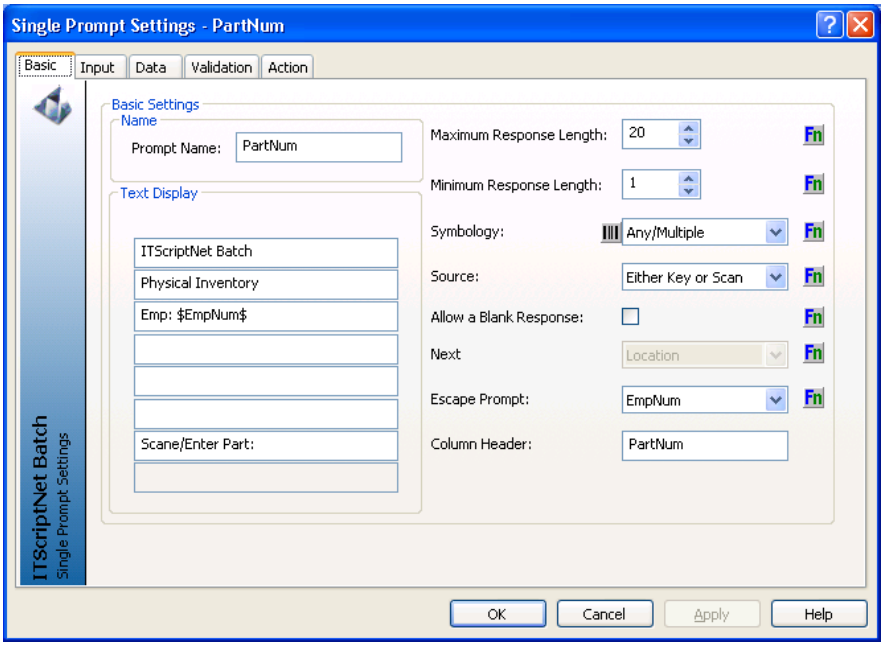
The Script Tree is located between the Prompt Flow area and the Main Design area. The Script Tree area in the design environment is optional. It can be turned on or off from the View Menu. The Script Tree area can also be resized. The main purpose of the Script Tree is to provide a view of your data collection program that shows all the prompts and any in-prompt scripts in a collapsible tree format for easy access. Please refer to the section in this User Guide on In-Prompt Scripts for

more information on scripts. The Script Tree also provides an alternate means of accessing your prompts and in-prompt scripts. Double-Click on either a prompt or a script in the Script Tree to bring up either the prompt's Settings screen or the In-Prompt Script Editor.

Single Prompt Settings

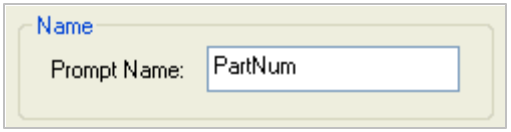
Each prompt has a set of properties, or settings. When a prompt is created, **IT*Script*Net** sets these properties to default settings, but each property for each prompt can be set during program design. Each prompt setting can also be determined during runtime with the use of In-Prompt scripts. Please refer to the In-Prompt Scripts section in this User Guide for more information. The Prompt Settings allow **IT*Script*Net** to have the *flexibility* to create almost any data collection solution.

The *Prompt Settings* screen can be accessed from the *Properties* menu item under the *Prompts* main menu. The *Prompt Settings* screen can also be accessed by right-clicking on a prompt and selecting *Properties…* from the pop-up menu. You can also double-click on the prompt in any of the areas in the main design environment to access the Prompt Settings screen.



Basic Tab

Prompt Name



The most basic property for a prompt is the prompt's name. The name you give to your prompt will be used throughout the program to refer to that prompt. The prompt name is displayed in the flowchart along the left side of the main application screen. The other prompts in the program may need to refer to the prompt in order to display information from that prompt or to set up the prompt's looping structure. It is a good idea to give the prompt a name that will reflect the prompt's data. The prompt name can be up to 20 characters and may contain letters or numbers. The following characters may not

be used on prompt names: # $ @ + - * . / = < > ( ) & as these characters are reserved for prompt substitution and variables (see Text Display section). The words 'Alias' and 'Timestamp' are also not permitted since using these as prompt names would conflict with the alias and timestamp information that is always captured for each record.

Text Display

The text display area on the *Prompt Settings* screen determines what will be displayed on the portable data terminal screen when the user is prompted to enter data. The number of lines available as text display will vary depending on the type of portable terminal currently configured for the program. The last line of the display is always reserved for the user to enter a response and is therefore not available as display area. You can enter any text that you wish to be displayed to the user so that the user has enough information to be able to input an appropriate response during data collection.

Prompt Substitution

There are three prompt substitutions that can be made in the Text Display area.

'$' Substitution

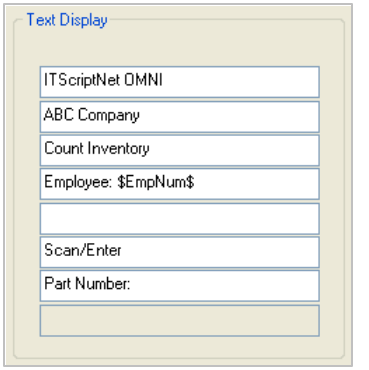
If you enter a prompt name surrounded by '$', i.e. '$EmpNum$', the terminal will substitute the last value collected for that prompt. Displaying results of previous prompts is a recommended program design strategy for creating user-friendly data collection solutions since it provides users easy reference to their previous responses during data collection

'#' Substitution

If you enter a prompt name surrounded by '#', i.e. '#EmpNum#', the terminal will substitute the last lookup file field for that prompt. Lookup fields are described in more detail in the section about the Validation Tab, below.

'@' Substitution

If you enter a variable name surrounded by '@', i.e. '@Desc@', the terminal will substitute the value of the user-defined variable. User-defined variables are described in more detail in the In-Prompt Scripts section of this User Guide.



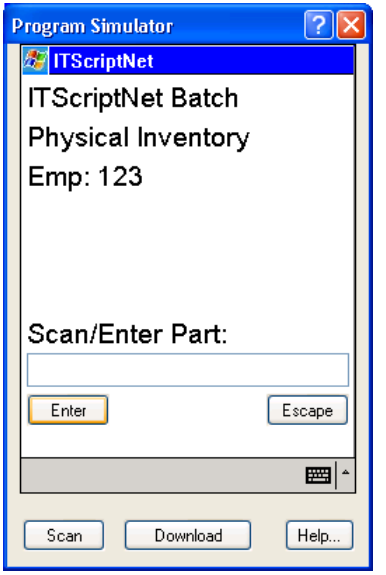
Examples:
In the *Prompt Settings* screen shown here, the first three lines of display text are simply text that identify the company and the data collection program.
The fourth line of display text is more complex. The "Emp: " portion of the fourth line is fixed text. However, the fixed text is followed by a special marker that will cause the program to display the response results of the prompt named within the $ markers. If during data collection, the user entered "ABC" as the data input response to the EmpNum prompt, then the fourth line of the display text for the PartNum prompt would read "Emp: ABC".

Text Effects

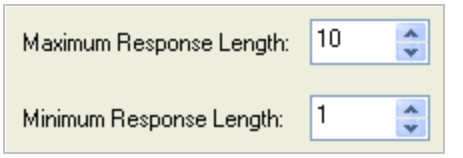
Control characters can be embedded in the text display area to change the appearance of the text. The table shows the control characters and the text effects available. The text effects that are available differ from terminal to terminal. As an example, if the "Emp: $EmpNum$" display text example were changed to "Emp: \V$EmpNum$", the result on the terminal would display the employee number data in bold and reverse and the user would see "Emp: 123".

Control Character	Text Display Effect	Terminals
\B	Bold	All non-DOS
\R	Reverse	All terminals
\V	Bold and Reverse	All non-DOS
\N	Normal	All terminals
\C0	Color-Black	WinCE / PocketPC / PC
\C1	Color-Blue	WinCE / PocketPC / PC
\C2	Color-Green	WinCE / PocketPC / PC
\C3	Color-Red	WinCE / PocketPC / PC
\C4	Color-Yellow	WinCE / PocketPC / PC
\C5	Color-Magenta	WinCE / PocketPC / PC
\C6	Color-Orange	WinCE / PocketPC / PC
\C7	Color-Teal	WinCE / PocketPC / PC
\C8	Color-Gray	WinCE / PocketPC / PC
\C9	Color-White	WinCE / PocketPC / PC

IT*Script*Net supports color text on WinCE and PocketPC devices, and the PC Client. There are additional Text Effects available to change colors in the same way that Bold and Reverse are selected. These text effects are listed in the table. You can combine Color effect with Reverse and Bold to achieve reverse or bold color.



Maximum and Minimum Response Lengths



The maximum and minimum response lengths are properties that set limits on the length of the response entered for the prompt. The limits of the number of characters for a response are the most basic form of validating data. For example, if each employee has a time clock number that is 3 or 4 digits, then the settings for the Employee clock number prompt can be set to require at least 3 characters/digits and no more than 4. When collecting data a user will not be able to enter a response longer than the maximum response length. If the user enters a response that is less than the minimum response length, the program will issue a warning to that effect, and the user will not be permitted to move to the next prompt unless the response is within the minimum and maximum response lengths. If the response is scanned, the program will still check the length of the scanned response and will reject any response that does not meet the response length criteria.

Symbology

For those devices on which **IT*Script*Net** supports the capabilities of the laser scanner or imager the Symbology property will be enabled. Consult the terminal specific sections supplied with this software to determine if this featured is supported for your device. The prompt setting for Symbology is used to limit the allowed barcode symbology when scanning a response to a prompt. There are many different barcode symbologies. A barcode symbology is an algorithm for encoding data into a barcode. Some of the more common symbologies are: Code 39, I2of5, Code 128, and UPCA. The prompt can be set to accept any barcode symbology (the terminal has to be able to decode the symbology, please refer to documentation on the specific portable terminal you are using for a list of symbologies compatible with your hardware). The Any/Multiple selection will allow a user to scan any symbology that is configured as a default for the scanner/imager for your hardware. The Any/Multiple setting for the Symbology prompt property is the default for **IT*Script*Net** for any new prompts that you create.

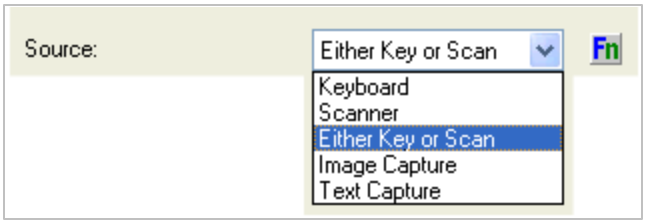
The Symbology prompt property can be set to accept only one type of barcode symbology. To select a single symbology, select it from the list. The default settings for the symbology on your terminal will be used. By way of example, let's say that your warehouse uses Code 39 barcodes for your part numbers. However, the boxes also contain UPC codes for those products. When collecting data, you want to insure that the Code 39 barcode is scanned and not the UPC code. By setting the prompt to only accept Code 39, the program will reject the UPC codes and any other symbology. Specifying a specific symbology is a means of validating your data and *maximizing data accuracy*.

Depending on your data collection needs, you may need to customize the Symbology prompt setting. In our example above, the warehouse had Code 39 and UPC part numbers. Let's say that now you need to be able to scan either a Code 39 or a UPC, but still do not want to allow any other symbologies. You can do this by clicking on the button with the barcode icon to bring up the *Advanced Barcode Settings* screen.

Please refer to the section in this User Guide describing the *Advanced Barcode Settings* screen. The *Advanced Barcode Settings* screen will also allow you to control each parameter available for one or more symbologies. If you do not customize the barcode symbology settings, the default settings for the one symbology specified or all symbologies will be used. The defaults are appropriate for most data collection applications, but if you need to control the settings more precisely, **IT*Script*Net** allows you access and full control of the symbologies. If the *Advanced Barcode Settings* screen is used to customize the symbology behavior for a prompt the barcode icon will change colors to provide an easy visual indicator that customized symbology settings are in use.

Source

For those devices in which ITScriptNet supports the capabilities of the laser scanner or imager the Source property will be enabled. Consult the terminal specific sections supplied with this software to determine if this featured is supported for your device. The prompt setting for source specifies whether the response to the prompt can be entered only on the terminal's keypad, only by scanning a barcode, or by either keypad or scan. By default, prompts are set to allow either keypad or scan for data entry. Depending on the application, requiring barcode scans will *reduce data collection time* (scanning is faster than punching in the data on a keypad) and *maximize accuracy*.



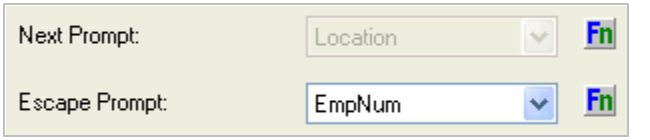
For data collection terminals equipped with an image reader to support image capture, two additional input sources are available: Image Capture and Text Capture. Image capture collects an image (in jpg or png format) for the prompt. Text capture is also an image capture, but optimized for capturing signatures or other text. Storing images as part of your collected data adds a new dimension for data collection. Image/text capture is essential for delivery confirmation applications, documenting damage, storing images of assets, or any application where a picture can be worth a thousand words.

Allow a Blank Response

This prompt setting will allow a blank response. By default, this setting is turned on meaning a response is not required to move on to the next prompt. If the box is not checked, the user is required to enter a valid response before moving on to the next prompt. Allowing a

blank response is useful for displaying messages and informational text to the user. Prompts that are used for optional information can be set to allow a blank response. As an example, a note field that allows a user to enter special notes would often be blank unless the user had a special circumstance that required keying in a note.

Next Prompt and Escape Prompt



The Next Prompt and Escape Prompt settings specify where a prompt fits into the overall prompting structure and program loop. When a response to a prompt is successfully entered, the program will proceed to the next prompt. If the user hits the Escape key instead, the program jumps to the Escape Prompt. The Next prompt is shown graphically with the green arrow and the Escape prompt is represented graphically by the red arrow. For every prompt except the last one, the Next Prompt setting cannot be changed and always shows the next prompt in the sequence. For the last prompt of the program this setting can be changed to control where the program loops after the last prompt is accepted. The Escape Prompt can be selected from the list of prompts. If no Escape Prompt is selected, the program will behave as if the preceding prompt has been specified as the Escape Prompt. Please refer to the section on Prompt Looping for more information.

Database Field/Column Header/Field Header

This property specifies where the data collected for this prompt is stored after it is sent to the PC and processed. Depending on the settings specified in the *Configure Receive* screen, this field may have different meanings.

If the program is configured to send its data to a Microsoft Access file or an ODBC data source, the target database field must be set up properly so the data collected can be sent to the desired field. A field from the target database must be selected from the list of fields. The fields in the list are the fields defined for the database and table that are established on the *Configure Receive* screen. The <Ignore> selection can be chosen and the collected data for this prompt will be ignored when the collected data is sent to the database. Fields are output to Access and ODBC if the field specified here exists and the maximum length of the prompt is not zero.

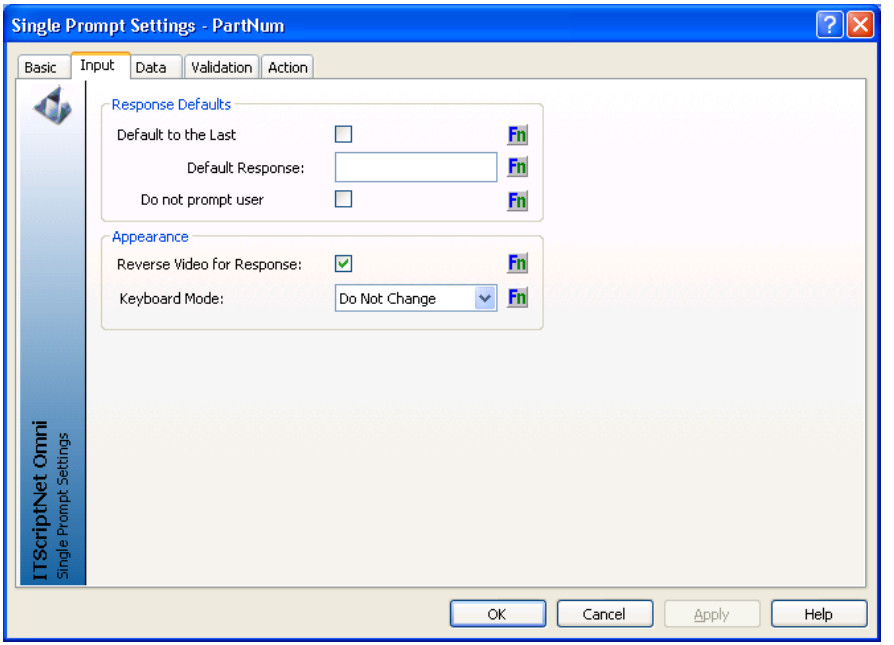
If the program is configured to send its data to a Microsoft Excel file, this setting will become the column header for the data. For Excel files, if this field is left blank, the Prompt Name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen.

If the program is configured to send its data to a text file with a header, this setting will become the field header for the data in the output file. For text files with headers, if this field is left blank, the Prompt Name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen.

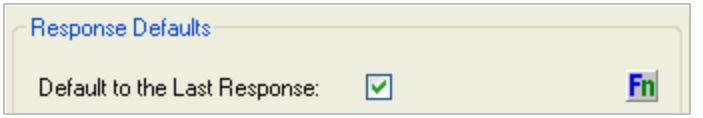
Input Tab

Response Defaults

The Response Defaults settings can set the default response to a specific value or the value of the response from the last data record collected. When creating a new prompt, the response default is initially set to no default. When in this state, the user will have to enter data for the prompt each time through the prompting loop during data collection.



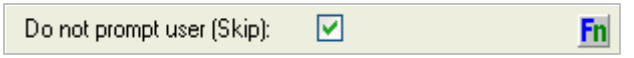
Using response defaults, the *Default to the Last Response* option can be enabled and will cause the response to default to the previous response. The user will still be able to change the response, but can save time by accepting the default. This feature is useful when a prompt will often have the same information for several prompting loops. For example, an asset physical inventory requires a location. The user would enter a location and then scan all the asset tags within that location before moving on to the next one. With the *Default to Last Response* option enabled, the location need only be entered the first time and then the field will default to that location until the user moves to another location and then enters a new location code.



The other option available for the response default is to actually set a response default that will always be shown as the default response for the prompt. The example shows a value of '100' as the default response for this prompt. This might be useful as an employee code, for example, in a situation where employee 100 is primarily responsible for collecting data with the program. Another employee could enter or scan his employee code to override the default, but having the default value saves time during data collection most of the time because the user simply can accept the default by pressing the Enter key.

Do Not Prompt User

There is a checkbox for the "Do Not Prompt User (Skip)" option. Normally, this box is unchecked and the prompt is displayed to the user. If checked, the prompt will be hidden (skipped). This is useful for certain special circumstances. One scenario where this option is useful is when a field of fixed data needs to be in the collected data file (to satisfy data processing), but it is not desired that the user be prompted for this data. Since each prompt represents a piece of collected data and vice-versa, the skipped prompt can have a default value set to a specific value that is required to be in the collected data file. One example of such an application would be if the collected data file represents transactions, and the transaction code is a required element in the data. The default value would be set to the transaction code, and the prompt would be skipped from the user's perspective. The transaction code would be able to be in the collected data file even though the user never input a response for that prompt.



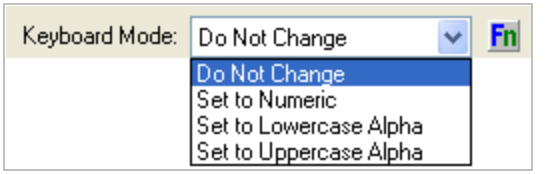
Appearance

Reverse Video for Response

The Appearance settings box has two properties. The first is the *Reverse Video for Response* property. If this option is checked, the last line of the screen for data collection will appear in reverse video. The default is to display the response area in reverse video. The reverse video option is only applicable to **IT*Script*Net** supported DOS-based terminals.

Keyboard Mode

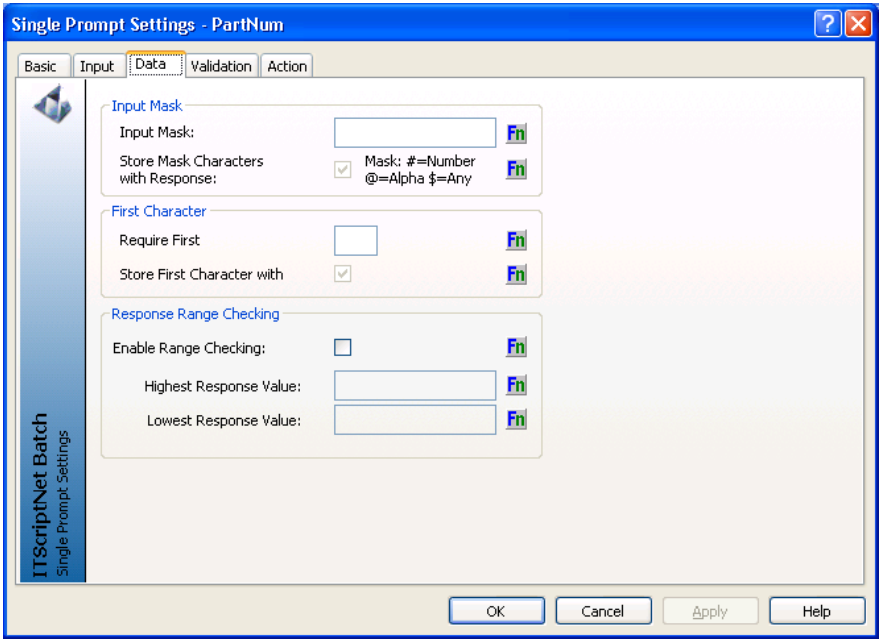
The *Keyboard Mode* property determines whether or not the keypad mode for the terminal will be set explicitly. If this property is *Set to Numeric*, **IT*Script*Net** will force the device into numeric data entry mode. Using this property is especially useful when designing data collection programs for devices that have keys that are combination alpha/numeric. If the prompt requires the user to enter a quantity, for example, automatically switching the terminal into numeric key mode will save the user time since the user will not have to switch the keypad into numeric mode. A prompt can also be configured to *Set to Uppercase Alpha* or *Set to Lowercase Alpha*, which are useful when the prompt's data is expected to be alphabetic. The default for this property is *Do Not Change*, which



means that **IT*Script*Net** does not change the keypad mode for the prompt. The options available in this list will depend on the terminal type selected for the program.

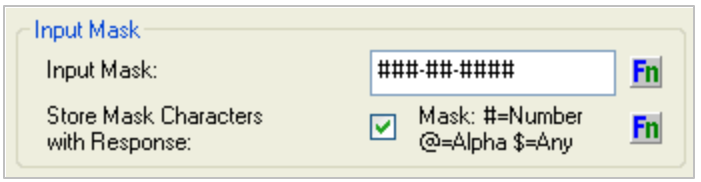
Data Tab

The Data tab has options for controlling the format that data must match. This allows the program to reject invalid data immediately and provide feedback to the operator.

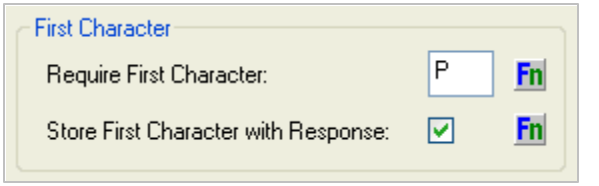


Input Mask

If the Input Mask is used for a prompt, the user will be required to enter data conforming to the input mask in order for the data to be accepted as a valid response to the prompt. The mask is entered as a string of characters. Any required character can be entered as well as any of the three wildcard placeholders in the mask. The # sign requires a number between 0-9 in that location in the input string. A @ symbol requires an alphabetic character (A-Z or a-z) and the $ is a placeholder for any character. In the example shown, the input mask is ###-##-####. This input requires 3 numeric characters, then a dash ('-'), then 2 more numeric characters, another dash ('-'), and then 4 numeric characters to complete the input. This input mask would correspond to a social security number. The input mask by default will store the whole input string, but by unchecking the 'Store Mask Characters with Response' box, the response can be stored without the specific mask characters. In our social security number format example, the number would be stored without the dashes if this option were turned off. **IT*Script*Net** has many data validation options in addition to the input mask. It is therefore important that all the validations be considered when designing the validation scheme for each prompt so that the validations can work together. If you specify an Input Mask, the Maximum and Minimum Length properties will automatically be set to match the length of the Input Mask and cannot be changed.

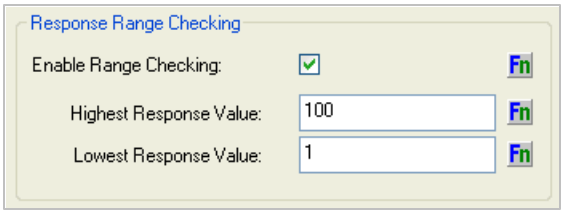


First Character



The First Character data validation feature allows the program designer to require the response to the prompt to start with the character specified. In the example shown, the letter 'P' is specified as the required first character. If the user scans a barcode or enters data that starts with something other than 'P', the response will be rejected. This feature is perfect for automotive industry labels that require part number barcodes to start with the letter P followed by the part number. The benefit of encoding a 'P' for part number or an 'L' for location (or any other key letter) within a barcode is that the user is protected from scanning the wrong barcode. This type of protection is especially useful if the label contains several different barcodes. The designer of the program has the option of specifying whether or not to store the first letter with the response. In the case of the part number following the letter 'P', it would often be advisable to remove the 'P' prior to storing the part number response so that the collected data can be processed without an extra step to remove the first character 'P'.

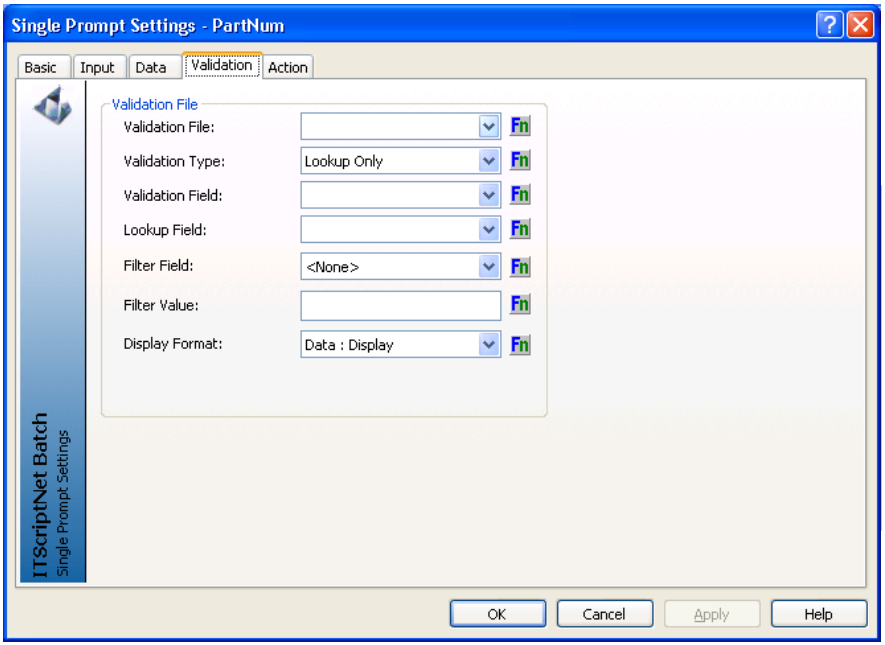
Response Range Checking



The Response range checking is another data validation feature of **IT*Script*Net**. When range checking is enabled, the response must be a numeric value. Negative and decimal values are permitted. The value of the response will be compared to the high and low response values. If the response entered is within the bounds of the range, the response will be accepted. If the response is higher than the highest response value or lower than the lowest response value set for the range, the response will be rejected and the user will get a message stating that the response was out of range. Enabling the range checking for responses is another way to *increase accuracy* of the data you collect.

Validation Tab

Validation Files



Validation files are text files that can be used to ensure that the operator enters accurate data. When the operator enters a prompt response, the entered data can be compared to the data in the validation file to make sure it is allowed. Additionally, a second field can be retrieved from the validation file and displayed to the operator. Typical uses might include:

- Part Number/Description lookups and validations
- Location Validation
- Employee Number Validation
- Order Picking
- Route/Delivery Lists
- Many More….

The validation text file must be defined from the *Validation Files* screen before a prompt can be configured to use the validation text file. Please refer to the *Validation Files* Screen section of this User Guide for more information on setting up the validation file for your data collection program.

For the purposes of this section, we will assume that a text file named Validate.txt has been setup for use by our data collection program. The text file contains a part number and a description.

Validation File

The drop-down selection box for the validation file will contain a list of all the validation files that have been identified for use by this program from the *Validation Files* screen. You can select which file you want to use to validate the response for this prompt.

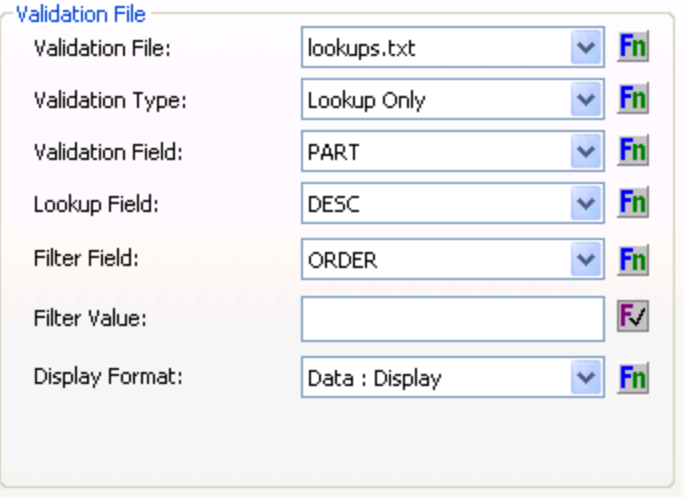
Validation Type

The second validation drop-down box contains the type of validation. There are four possible validation types.

- **Lookup Only**
  This is used to perform a lookup, but does not require the response to exist.
- **Must Be Found**
  This mode means that the entered data must exist in the lookup file. If the response entered is not found in the validation file, the response is rejected.
- **Must Not Be Found**
  This mode means that the entered data must NOT exist in the lookup file. If the response is already in the lookup file the response is rejected.
- **Pick From List**
  The *Pick From List* validation type causes the prompt to present the user with a list of choices to choose from instead of entering the data. The Validation field and the Lookup field are the data displayed to the user on the terminal as a list.

Validation Field

The third drop-down selection is the field name to be used to validate this prompt. When the Validate.txt file was added to the list of validation files for this program, the definition for that file was also configured. One of the fields in the file was named 'Part' and the description included in the validation was named 'Desc'. In our example, this prompt that we are validating is the part number so the Part field is selected as the Validation Field.



Lookup Field

The description, which has been defined to be in the Desc field, is selected in the fourth drop-down selection as the information to lookup as the result of the validation. The description can be displayed to the operator later by using the # marker in the display text for a later prompt. The Lookup data is not stored in the collected data file. Only the entered responses are stored in the collected data file.

In the special case of the picklist, the user is presented with a list consisting of both the Validation Field and the Lookup Field. When the user selects his choice from the list, the value of the Validation Field (in our example, the Part number) is stored as the response to the prompt. Using In-Prompt Scripts (See the *In-Prompt Scripts* Section of the User Guide), the picklist information

can be customized to display other fields in addition to (or instead of) the Validation Field and the Lookup Field, which is the standard picklist behavior.

Filter Field

The Filter Field selection applies only to Picklists. It is used to apply a filter to the validation file so that only matching records are added to the picklist. For example, if you were filling a Picklist from an Order Items file, you might specify an Order Number field as the filter field, and a specific order number as the Filter Value so that only items on that order are listed in the Picklist. Select one of the fields from the validation file, or <None> if you do not want to filter the file.

Filter Value

This is the value that should be used to filter the validation file. You can enter a value, or use the In-Prompt Script to override the Filter Value so you can filter by the value of another prompt or variable.

Display Format

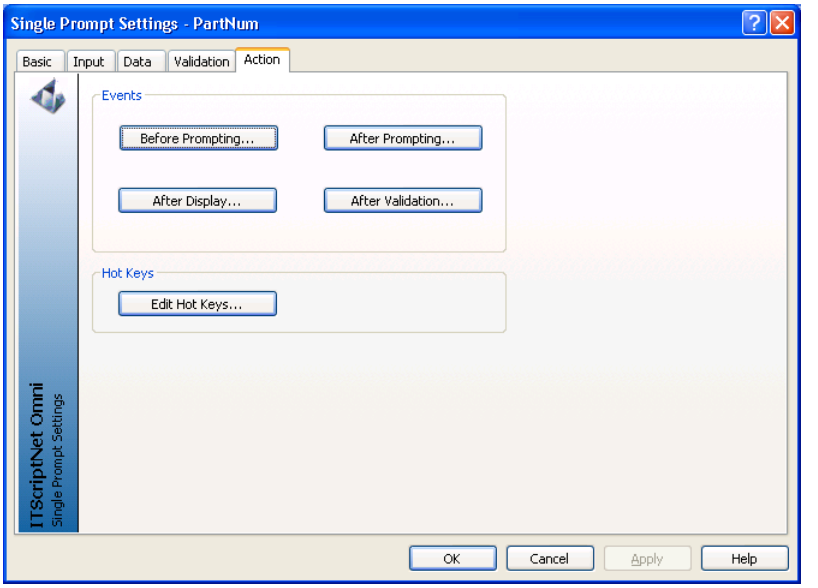
This field allows you to control how data is displayed in the Picklist. The standard format is to display both the Data (the Validation Field) and the Display (the Lookup Field), in the format Data : Display. You can also specify to display just the Data field, or just the Display field.

Override Display Field

This In-Prompt script is only available for Picklists. This script allows you to control the way data is presented in the picklist even further than the Filter and Display options already described. This script is executed once for each record in the validation file. The result of the script is used as the text to display in the picklist, overriding the default display. If the script returns an empty string, the record will not be added to the picklist. You can reference the fields in the validation file record using the *PickListField* function. This allows you to completely control the way data is presented in a Picklist.

The picklist can also be combined with the default prompt setting. If the prompt is configured to use a picklist and a default value is used, the default value will be the default selection in the picklist. This combination will allow the user to save time on the prompt by pre-selecting the most-common selection from a picklist.

Action Tab



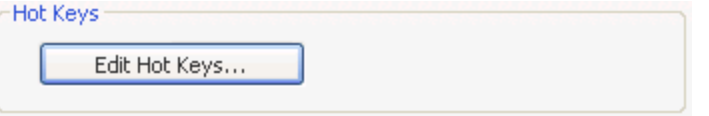
The Action tab contains the settings to control the behavior of the prompt before it is displayed, after it is displayed, after data has been entered, and after the data has been validated.

Events

The Events area allows you to include In-Prompt Scripts for the prompt. In-Prompt scripts allow additional functionality to be included in your programs. The Before Prompting script will execute before the prompt is displayed. This script is especially useful for performing lookups or calculations that will be displayed to the user. The After Display script runs immediately after the prompt is displayed to the user. The third In-Prompt script is the After Prompting script. It runs after the user has performed an action that would cause the program to go to the Next prompt. The After Prompting script runs after the user takes the action that would cause the program to go to the Next prompt but before the built-in validations for each of the elements on the prompt run, such as whether or not to allow a blank response. Lastly, after the built-in validations run, the After Validation In-Prompt script is run. This script allows a final opportunity to check for any undesirable data, perform any lookups, or do any calculations prior to leaving the prompt and moving to the next prompt. Please refer to the section on In-Prompt scripts for more information.

Hot Keys

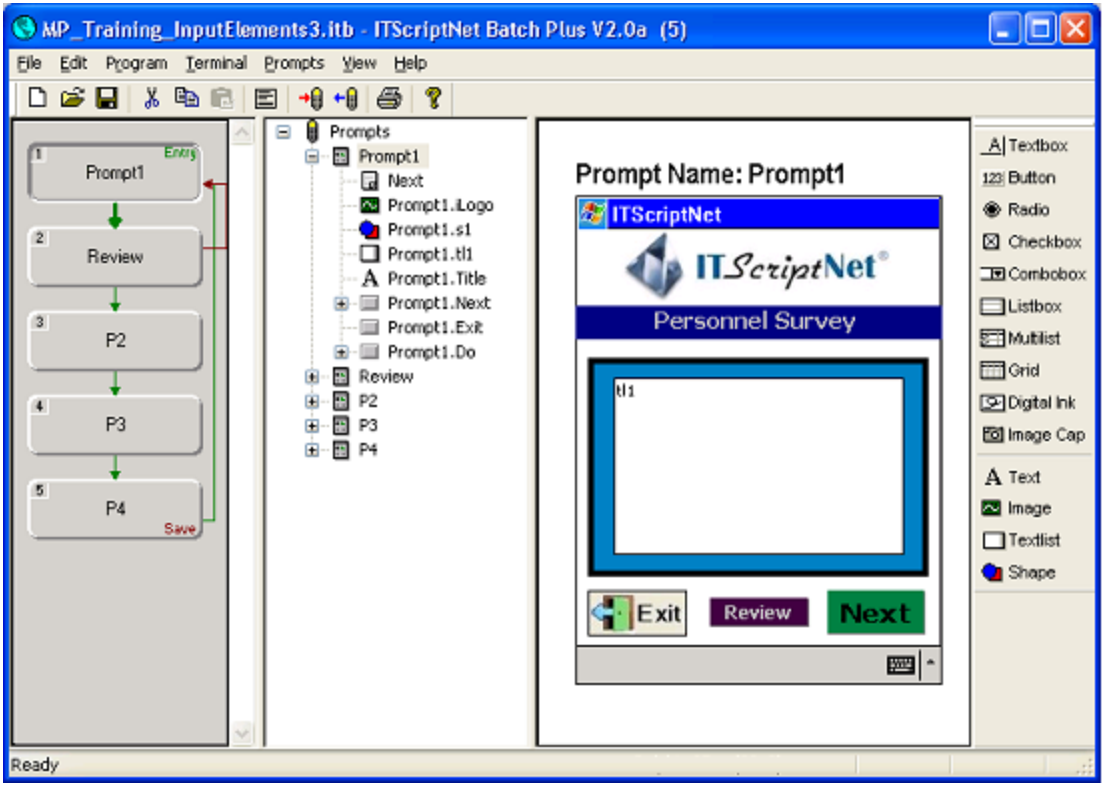
The Hot Keys button allows you to specify Hot Keys that are available on the prompt. For more information, see the *Hot Key Editor* section of the User Guide.



Multi-Prompt

Design Environment

This feature is only available on devices with Windows CE or Pocket PC operating systems. A Multi-Prompt is a prompt in **IT*Script*Net** that contains one or more elements. **IT*Script*Net** has a rich set of elements that can be used to display information to a user and/or obtain input from the user. While a Single Prompt has one piece of collected data per prompt, a multi-prompt can have many elements that collect data per prompt. The **IT*Script*Net** design environment contains several sections, each geared towards helping you design data collection programs.



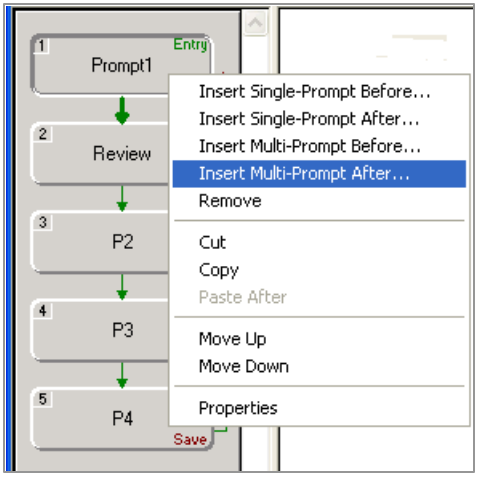
Prompt Flow

The left side of the main window is the prompt list. The prompt list is like a flow chart showing the order of the prompts. Each prompt is listed. There is a green arrow connecting the prompts showing the path from a prompt to the prompt specified as the Next Prompt. The Next Prompt is the prompt that the program advances to after the user has correctly entered data for all applicable

elements on a prompt. Red arrows show the path the prompt specified as the Escape Prompt. The Prompt area allows you to control several prompt-level actions.

Creating a Multi-Prompt

To create a new prompt, right-click on a prompt and select the Insert Multi-Prompt After or Insert Multi-Prompt Before menu item from the popup menu. You can also select one of these Insert menu items from the Prompts main menu.



Copy a Multi-Prompt

You can also create a new prompt by right-clicking on a prompt, selecting copy, right-clicking on the same or another prompt and selecting Paste After. You will be prompted to enter a name for the new prompt. Copying prompts can save time since all of the elements, settings and scripts for the prompt will be copied to the new prompt.

Remove a Prompt

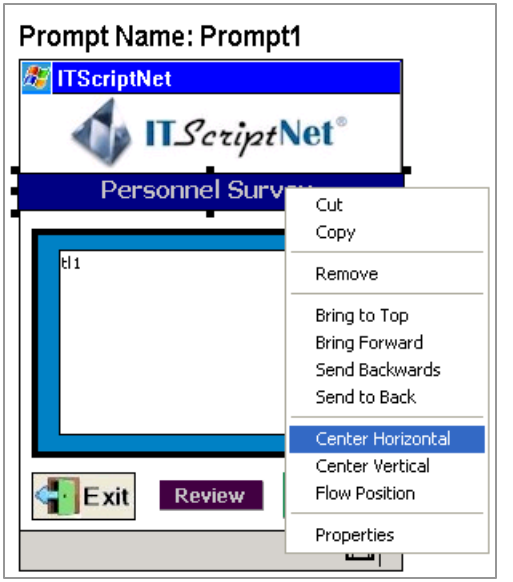
Prompts can be deleted by right-clicking on a prompt and selecting the Remove menu item. You will be asked to confirm deleting the prompt since removing a prompt cannot be undone. The Remove menu item is also available from the Prompts main menu.

Move Up/Move Down

You can move prompts relative to each other with the Move Up and Move Down menu items. Access the Move Up and Move Down menu items from either the right-click menu or from the Prompts main menu item.

Properties

The Properties menu item will bring up the Multi-Prompt Settings screen. This screen defines the properties, or settings, for the prompt. Each of these settings is described in a later section.



Main Design Area

For Multi-Prompts, the main design area provides a visual representation of the prompt. The representation of the prompt will be slightly different depending on the terminal type. This area also is an interactive area for modifying the prompt. You can accomplish many types of tasks in the design area. Typically, a multi-prompt will have several

elements. You can see each of the elements for the prompt on the design area. If you select one element or multiple elements and right-click, you can access several element-level actions. These element-level actions are described below.

Cut, Copy, Paste, Remove

Cutting an element will remove it from the prompt, but the element will be available to right-click again and use the Paste menu item. You can also copy an element to create multiple elements that are the same (except for the names of the elements). Copy and Paste can be an efficient design strategy to minimize setting edits if you need to create elements that are similar. You can also copy and paste an element from one prompt to another as well as within a prompt. The Remove menu item deletes the element from the prompt and does not make the element available for pasting. The Remove action on an element can be undone with the Undo menu item.

Bring To Top, Bring Forward, Send Backwards, Send to Back

These menu items control how the selected element is displayed relative to other elements which overlap the selected element. Bring to Top will layer the selected element over top of all other elements so it will be 'on top'. The Send to Back menu item will send the selected element all the way to the back so that any other elements will be on top of the selected element. Move Forward and Move Backward move the selected item one step at a time in the order respectively.

Center Horizontal

The Center Horizontal menu item will position the selected element within the prompt such that it is centered from side to side. There will be an equal amount of space on either side of the element.

Center Vertical

The Center Vertical menu item will position the selected element within the prompt such that it is centered up and down. The element will be centered on the prompt such that there will be an equal amount of space on the top and bottom of the element.

Flow Position

The flow position feature assists in obtaining consistent positioning and sizing for elements that have the same name but are on different prompts. If you use elements on several prompts you can reposition the element on one prompt and then use the Flow Position menu item to make the new position and size flow to all the other elements that have the same element name but are on the other prompts. It is typical to use some design elements such as logos, titles, buttons, etc. on several prompts in a data collection program.

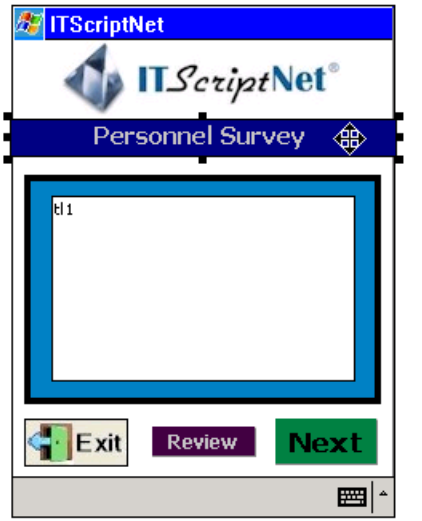
Properties

The Properties Menu Item will bring up the properties, or settings, screen for the selected element. The element types and each of the settings for each element type is described later in this User Guide.

Resize Element

You can resize an element by selecting it and then positioning the cursor over the resize blocks. Your cursor will change to the resize cursor and you can then drag the edge of the element to resize it.

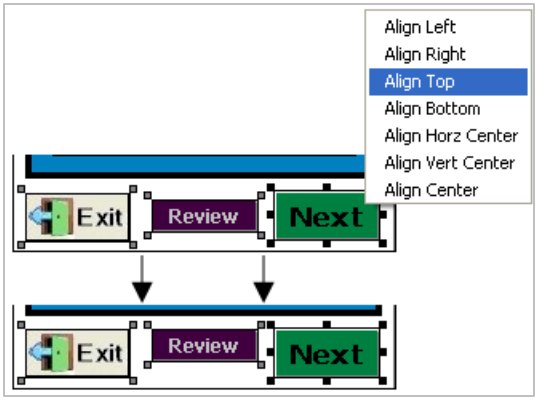
Move Element



You can move an element by selecting the element and then dragging the element to the new position. You will see the move cursor when your mouse cursor is over a selected element to move. You can also move a group of objects all at once if you multi-select elements. Another method for moving elements is to select one or more elements and then use the keyboard arrow keys to nudge the position one pixel at a time in the direction of the arrow key.

Element Alignment

There are several alignment menu items available when more than one element is selected. Click on the first element to select it and then hold down the Ctrl key on the keyboard while clicking the other elements you wish to select. Another way to multi-select elements is to use the 'rubber band'. Drag the mouse with no item selected across the design area to select elements touching the region enclosed by the mouse rectangle. Once you have the elements selected, right-click the mouse and you will see many available multi-element actions on the pop-up menu. The alignment options include: Align Left, Align Right, Align Top, Align Bottom, Align Horiz Center, Align Vert Center, and Align Center. Each of these alignment options will use the element outlined with the solid black resize rectangles as the anchor element and all other elements selected will be repositioned to align to the anchor. In the example shown, the three button elements have been selected. The button on the right has the black resize rectangles to indicate that it is the anchor element. After clicking on Align Top the other 2 elements are re-positioned so that the tops of all selected elements are even with the anchor elements.

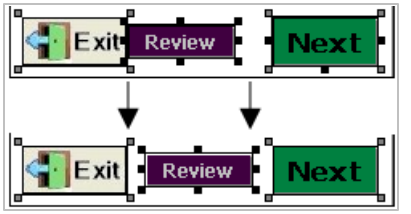


Same Height, Same Width, Same Both

These 3 menu options are available when more than one element is selected. Like the alignment options, these features will use the anchor element as the basis for the action. The *Same Height*

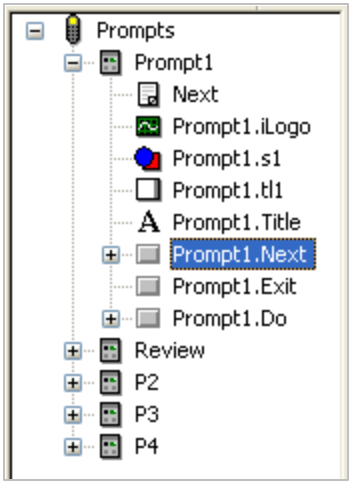
menu option will make all selected elements equal in height to the anchor element. The *Same Width* will make all the selected elements the same width and the *Same Both* option will make all the selected elements the same size in both directions.

Space Evenly Down, Space Evenly Across



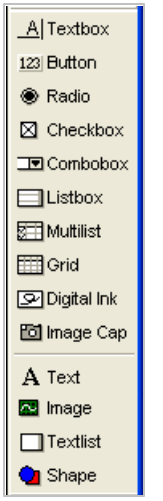
These options will position the selected elements so that there is even spacing between them. For the Space Evenly Across option, the left-most and right-most elements will remain in their original position. All other selected elements will align themselves so that the spacing left-to-right is equal between the selected elements. Space Evenly Down works in a similar manner except the Top-most and bottom-most elements will remain fixed and the other elements will align themselves so that there is even spacing vertically between the elements.

Script Tree



The Script Tree is located between the Prompt Flow area and the Main Design area. The Script Tree area in the design environment is optional. It can be turned on or off from the View Menu. If you do not see the Script Tree area, make sure that the Script Tree menu option has a checkmark next to it in the View Menu. The Script Tree area can also be resized. The main purpose of the Script Tree is to provi de a view of your data collection program that shows all the prompts and elements and in-prompt scripts in a collapsible tree format for easy access. Please refer to the section in this User Guide on In-Prompt Scripts for more information on scripts. The Script Tree also provides an alternate means of accessing your prompts and elements. Double-Click on either a prompt or an element in the Script Tree to bring up either the prompt's or element's Settings screen.

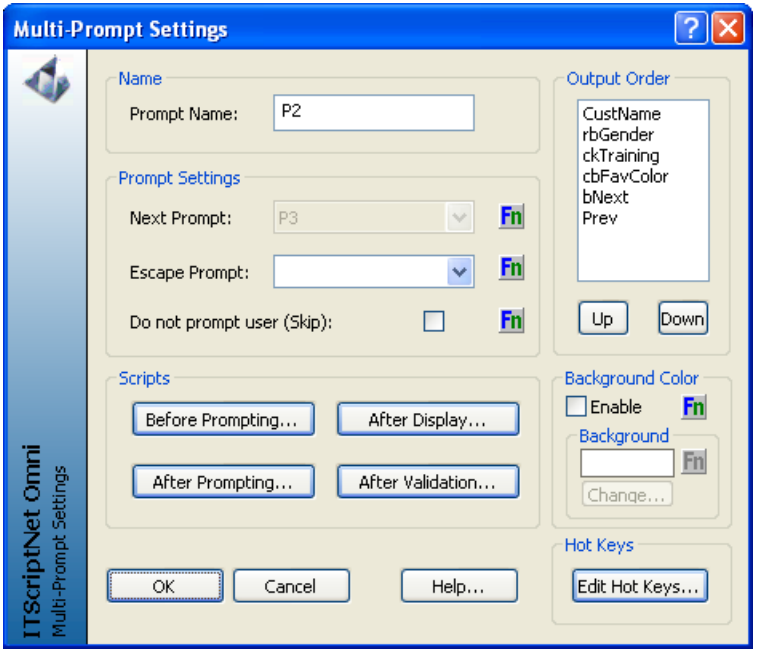
Elements Area



The Element Toolbar is located on the right side of the design environment. It is a list of elements available for multi-prompts. This toolbar can be turned on or off from the View Menu. If you do not see the Elements toolbar, make sure that the Elements menu option has a checkmark next to it in the View Menu. The elements at the bottom are display elements which do not collect data, but enhance the prompt by displaying information to the user or by creating an appealing user interface. The display elements are: Text, Image, Textlist, Shape, and Timer. The Button is unique in that it accepts input from the user (the user clicks the button), but it does not have collected data associated with it. The remaining elements are inputs and when used on a prompt can collect data from the user: Input Textbox, Radio Button, Checkbox, Combobox, Listbox, Multilist, Grid, Digital Ink, and Image Capture. Each Element is described in detail in its own section to follow.

Multi-Prompt Settings

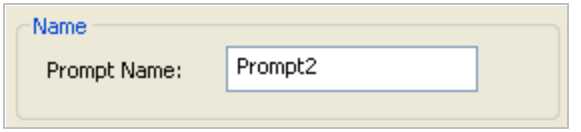
Each prompt has a set of properties, or settings. When a prompt is created, **IT*Script*Net** sets these properties to default settings, but each property for each prompt can be set during program design. Each prompt setting can also be determined during runtime with the use of In-Prompt scripts. Please refer to the In-Prompt Scripts section in this User Guide for more information. The Prompt Settings allow **IT*Script*Net** to have the *flexibility* to create almost any data collection solution.



The *Multi-Prompt Settings* screen can be accessed from *Properties* under the *Prompts* main menu. The *Multi-Prompt Settings* screen can also be accessed by right-clicking on a prompt and selecting *Properties…* from the pop-up menu. You can also double-click a prompt in either the Prompt Flow or Script Tree areas in the design environment to access the *Multi-Prompt Settings* screen.

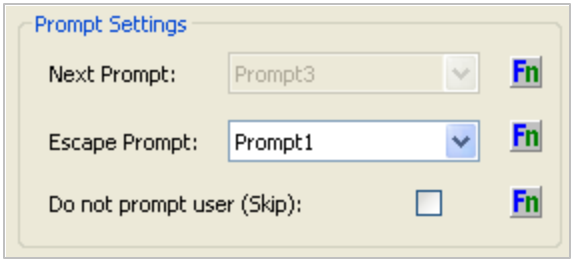
Prompt Name

The most basic property for a prompt is the prompt's name. The name you give to your prompt will be used throughout the program to refer to that prompt. The prompt name is displayed in the flowchart along the left side of the main application screen. The other prompts in the program may need to refer to the prompt in order to display information from that prompt or to set up the prompt's looping structure. It is a good idea to give the prompt a name that will reflect the prompt's data. The prompt name can be up to 20 characters and may contain letters or numbers. The following characters may not be used on prompt names: # $ @ + - * . / = < > ( ) & as these characters are reserved for prompt substitution and variables. The words 'Alias' and 'Timestamp' are also not permitted since using these as prompt names would conflict with the alias and timestamp information that is always captured for each data record.



Next Prompt and Escape Prompt

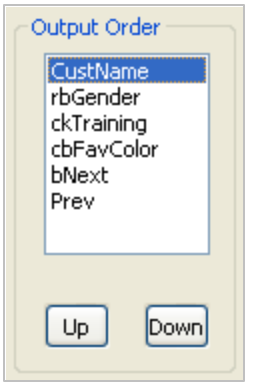
The Next Prompt and Escape Prompt settings specify where a prompt fits into the overall prompting structure and program loop. When responses to the elements on multi-prompt are successfully entered, the program will proceed to the next prompt when the user performs an action that causes the program to advance, such as clicking a Next button. If the user performs an action to cause the prompt to escape instead, such as clicking an Escape button, the program jumps to the Escape Prompt. The Next prompt is shown graphically with the green arrow and the Escape prompt is represented graphically by the red arrow. For every prompt except the last one, the Next Prompt setting cannot be changed and always shows the next prompt in the sequence. For the last prompt of the program this setting can be changed to control where the program loops after the last prompt is accepted. The Escape Prompt can be selected from the list of prompts. If no Escape Prompt is selected, the program will behave as if the first prompt has been specified as the Escape Prompt. Please refer to the section on Prompt Looping for more information. The Next and Escape Prompt settings can also be used with In-Prompt scripts to achieve conditional branching. Please see the section in this User Guide about In-Prompt Scripts for more information about conditional branching.



Do Not Prompt User (Skip)

There is a checkbox for the "Do Not Prompt User (Skip)" option. Normally, this box is unchecked and the prompt is displayed to the user. If checked, the prompt will be hidden (skipped). This option can be an effective means of controlling program flow, especially if the value of this setting is determined by an In-Prompt Script. Please see the section in this User Guide about In-Prompt Scripts and how they are used to achieve conditional branching.

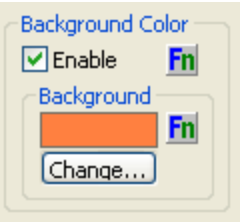
Output Order



The Output Order list is a list of all the input elements and action button elements on the Multi-Prompt. The order of the list determines the tab order. When the user first sees a prompt, the focus will be on the first element in this list. The element with the focus is the element that is ready to receive input. If the user presses the tab button, the focus will move to the next element in the list. When the last element in the list gets the focus, the next tab action will move the focus back up to the first element. The element with the focus can also be changed with in-prompt script functions or other element settings, so the tab order does not necessarily follow the order shown in the list.

The output order is also used when processing the collected data as text files or Excel files. The order of the fields in the output file will be as listed in the Output Order unless modified with the *Customize Field Layout* screen. Please refer to the *Customize Field Layout* section in another section of this manual.

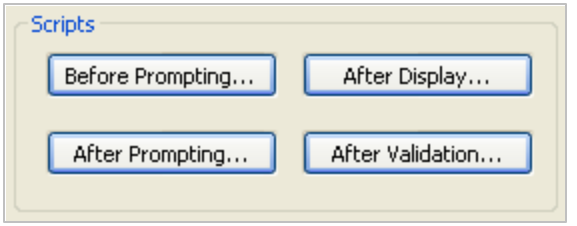
Background Color



The background color setting can enhance the user interface and is really only effective on terminals with a color display. To use a background color, check the Enable option. Once enabled, the Background color can then be changed by clicking on the Change… button and selecting a color from the color screen.

Scripts

The scripts area defines In-Prompt Scripts for the prompt. In-Prompt scripts allow additional functionality to be included in your programs. The Before Prompting script will execute before the prompt is displayed. This script is especially useful for performing lookups or calculations for elements that will be displayed to the user. The After Display script runs immediately after the prompt is displayed to the user. It is a good place to add functions that control the element that has the focus. The third In-Prompt script is the After Prompting script. It runs after the user has performed an action that would cause the program to go to the Next prompt. The After Prompting script runs after the user takes the action that would cause the program to go to the Next prompt but before the built-in validations for each of the elements on the prompt run, such as whether or not to allow a blank response for an element. After the built-in validations run for all the elements on the prompt, the last thing that happens before the program flow transfers to the Next Prompt is the After Validation In-Prompt script. This script allows a final opportunity to check for any undesirable data, perform any lookups, or do any calculations prior to leaving the prompt and moving to the next prompt. Please refer to the section on In-Prompt scripts for more information.

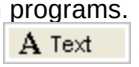


Hotkeys

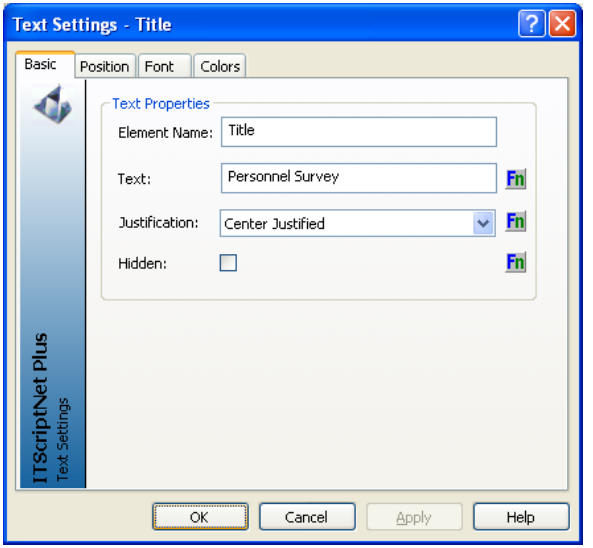
Press the Hotkeys button to configure prompt-specific Hotkeys. For more information on editing Hotkeys, see the **Hotkey Editor** section of this manual. Any Hotkeys you define here will be effective only for this prompt. If you create a Hotkey for a prompt that is the same as a global Hotkey, the Prompt Hotkey will override the global one on that prompt.

Text Element

The Text element is an element for displaying informational text to a user. The Text element can be customized to enhance the look-and-feel of data collection programs.

To add a Text element to a prompt, click on the Text element 

from the element list on the right side of the main Generator Application screen. If you cannot see the element list, you can turn it on by checking the Elements menu under the View main menu item. After you click the Text element, the Text Settings screen will be displayed. You will need to specify the settings for your Text element. The key settings are the element Name and the text to display. These and all Text Settings are discussed below.



Basic Tab

The Basic tab on the Text Settings screen is shown here.

Element Name

Text elements, like all elements, must have a name. When you first create the Text element the name will be displayed as “Text1”. You may change the element name to another valid element name if you wish. The name of the element can be up to 20 characters in length and must be a unique name within the prompt. The following characters may not be used in element names: # $ @ _ - * . / = < > ( ) & as they are reserved.

Text

This is the text to be displayed to the user.

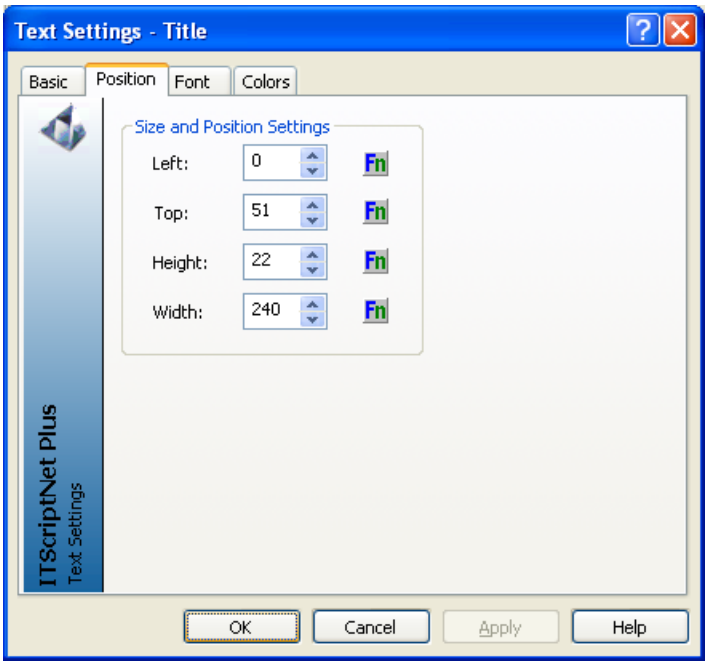
Justification

This setting will determine whether the text is left, right, or center justified within the text element.

Hidden

The Hidden setting will cause the Text element to be hidden when checked.

Position Tab



The Position tab on the Text element screen controls the size and position of the Text element on the prompt. You can also control the size and position of the Text element from the main design area. You can move the Text element by selecting it and holding the mouse while you move the Text element to the desired location. You can also resize the Text element by selecting it and hovering over the resize boundaries of the Text element and dragging to resize it.

Left Position

The Left Position setting specifies the horizontal location in pixels of the upper-left corner of the Text element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Left Position setting is set to 0, the Text element will be all the way to the left of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the left position to a value larger than the screen the element will be located off the prompt screen.

Top Position

The Top Position setting specifies the vertical location in pixels of the upper-left corner of the Text element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Top Position setting is set to 0, the Text element will be all the way to the top of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the top position to a value larger than the screen the element will be located off the prompt screen.

Height & Width

The Height setting specifies the height in pixels of the Text element and the Width setting specifies the width in pixels of the Text element.

Font Tab

The Font tab on the Text Settings screen allows you to customize standard Text elements.

Font

The Font selection allows you to choose a font for the text. Note that if you select a font that is not installed on your portable terminal, the operating system of the terminal will convert the font to one that is installed and the size may not exactly match the design. The area under the font selection displays a preview of the font you have chosen.

Attributes

The Attributes section allows you to include Bold, Italic, Underline, or Strikeout attributes to you font choices. A check next to the attribute turns that attribute on.

Height & Width

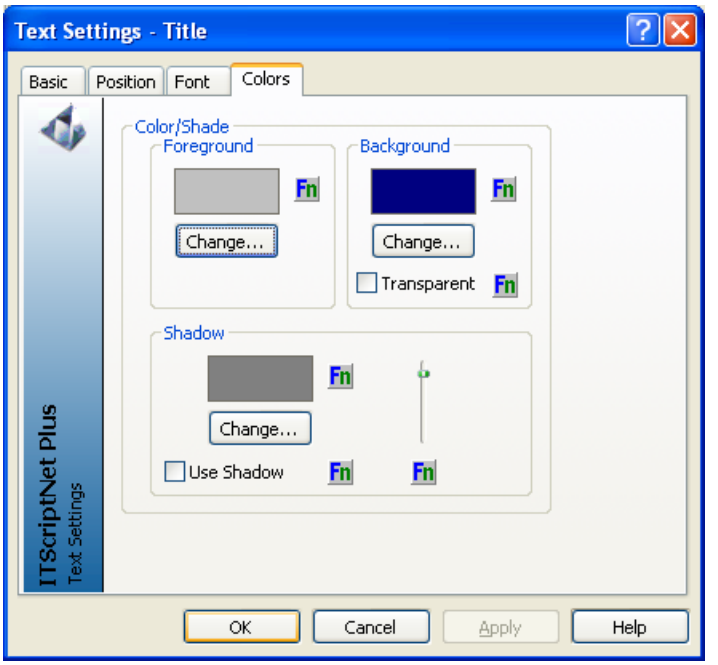
The Height and Width font settings specify the Height and Width of the font size in pixels. The Font Preview section will reflect the Height and Width specified. If you use a Font Width of zero, Windows will select an appropriate default width.

Style

The Style setting allows you to apply a pre-defined Font and Color set to this element. The Style must have been created with the Style Editor on the Program menu. Any changes made to the Style in the Style Editor will automatically be applied to this element. If you change any Font or Color setting on the element, the Style will be removed.

Colors Tab

The Colors tab adds another dimension of flexibility and customization to the design of Text elements. The colors will only be effective for portable terminals that have a color display.

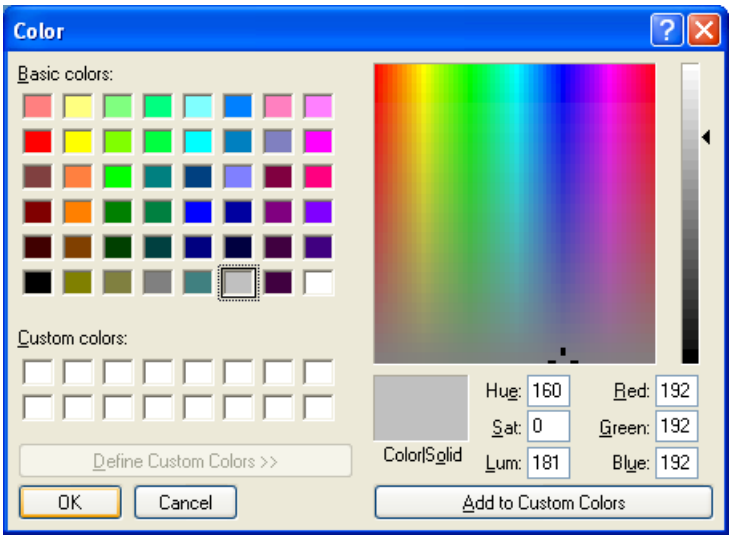


Foreground and Background Color

The Background color and the Foreground (Text) color on the Text element can be specified independently. Click on the Change button to bring up the Color screen. You can select from the basic colors on the left, select a color from the color matrix on the right, specify Hue/Saturation/Luminosity, or specify Red, Blue, and Green values to specify a custom color.

Shadow

The Shadow settings allow you to determine if the text should use a shadow and if so, how large the shadow should be. The shadow distance determines the number of pixels the shadow is positioned away from the main text. The color to use for the shadow can also be changed by clicking the change button and then selecting a shadow color.

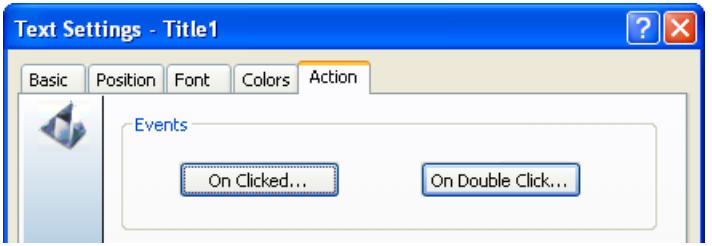


Color Screen

The color screen is used throughout **IT*Script*Net** for making color selections. You can select from the basic colors on the left, select a color from the color matrix on the right, specify Hue/Saturation/Luminosity, or specify Red, Blue, and Green values to specify a custom color.

Action Tab

The Action tab is used to control the elements response to Events.



Events

The Text element has several special event-driven scripts that allow for customized behavior.

- The **OnClick** Script runs whenever the element is tapped or clicked.
- The **OnDoubleClick** Script runs whenever the element is double-tapped or double-clicked.

Textlist Element

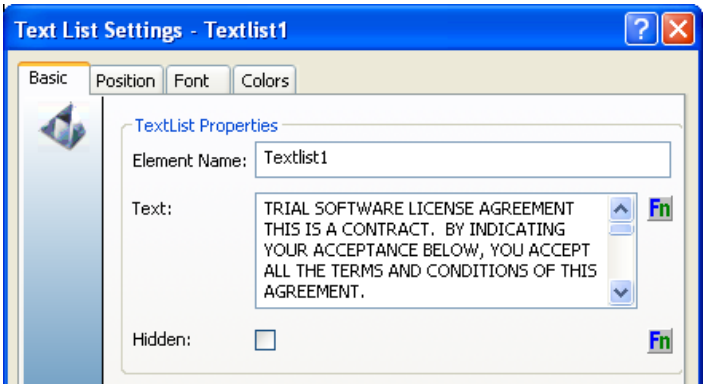
The Textlist element is an element for displaying informational text to a user. The Textlist element is very similar to the Text Element. The main difference is that the Textlist element can be scrolled to display longer text than the Text Element.

To add a Textlist element to a prompt, click on the Textlist element 

from the element list on the right side of the main Generator Application screen. If you cannot see the element list, you can turn it on by checking the Elements menu under the View main menu item. After you click the element button, the Textlist Settings screen will be displayed. You will need to specify the settings for your new element. The key settings are the element Name and the Textlist to display. These and all settings are discussed below.

Basic Tab

The Basic tab on the Textlist Settings screen is shown here.



Element Name

Textlist elements, like all elements, must have a name. When you first create the Textlist element the name will be displayed as “Textlist1”. You may change the element name to another valid element name if you wish. The name of the element can be up to 20 characters in length and must be a unique name within the prompt. The following characters may not be used in element names: # $ @ _ - * . / = < > ( ) & as they are reserved.

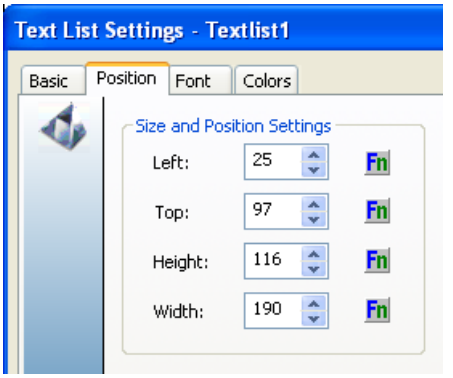
Text

This is the Text to be displayed to the user.

Hidden

The Hidden setting will cause the Textlist element to be hidden when checked.

Position Tab



The Position tab on the Textlist element screen controls the size and position of the Textlist element on the prompt. You can also control the size and position of the element from the main design area. You can move the Textlist element by selecting it and holding the mouse while you move the element to the desired location. You can also resize the Textlist by selecting it and dragging the resize rectangles.

Left Position

The Left Position setting specifies the horizontal location in pixels of the upper-left corner of the Textlist. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Left Position setting is set to 0, the Textlist element will be all the way to the left of the prompt.

Top Position

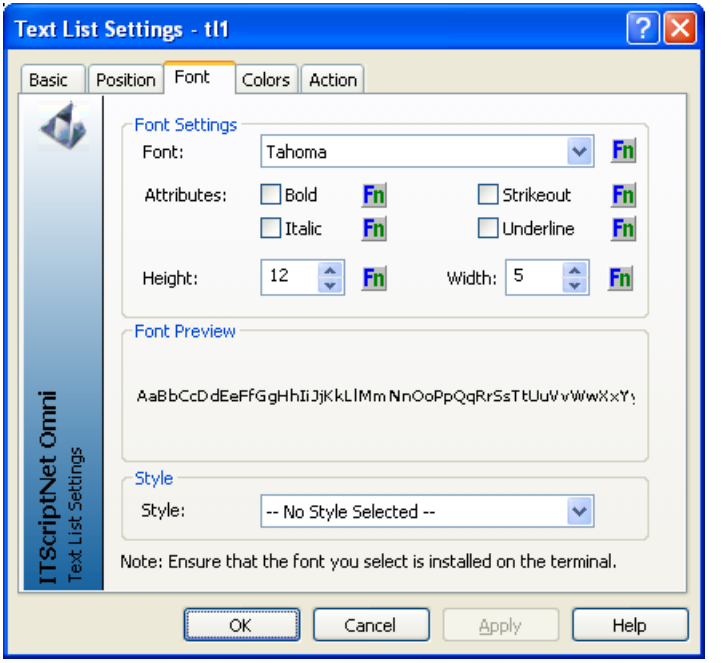
The Top Position setting specifies the vertical location in pixels of the upper-left corner of the Textlist element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Top Position setting is set to 0, the Textlist element will be all the way to the top of the prompt

Height & Width

The Height setting specifies the height in pixels of the Textlist and the Width setting specifies the width in pixels of the Textlist.

Font Tab

The Font tab on the Textlist Settings screen allows you to customize standard Textlist elements.



Font

The Font selection allows you to choose a font for the Textlist. Note that if you select a font that is not installed on your terminal, the operating system of the terminal will convert the font and the positioning may not be consistent with your design. The area under the font selection displays a preview of the font you have chosen.

Attributes

The Attributes section allows you to include Bold, Italic, Underline, or Strikeout attributes to your font choices. A check next to the attribute turns that attribute on.

Height & Width

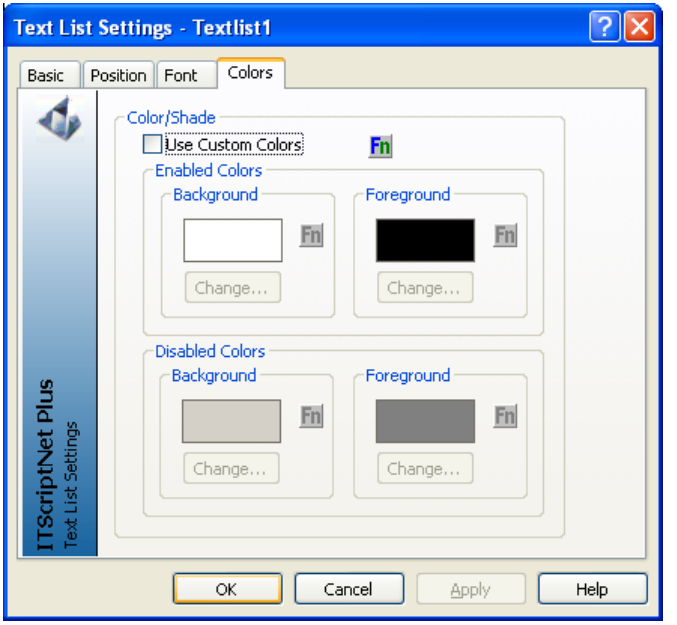
The Height and Width font settings specify the Height and Width of the font size in pixels. The Font Preview section will reflect the Height and Width specified. If you use a Font Width of zero, Windows will select an appropriate default width.

Style

The Style setting allows you to apply a pre-defined Font and Color set to this element. The Style must have been created with the Style Editor on the Program menu. Any changes made to the Style in the Style Editor will automatically be applied to this element. If you change any Font or Color setting on the element, the Style will be removed.

Colors Tab

The Colors tab adds another dimension of flexibility and customization to the design of elements. The colors will only be effective for portable terminals that have a color display. The first step towards customizing the colors of a standard button is to check the 'Use Custom Colors' option on the Colors tab. Once checked, the colors specified on the tab will be used.



Enabled Colors

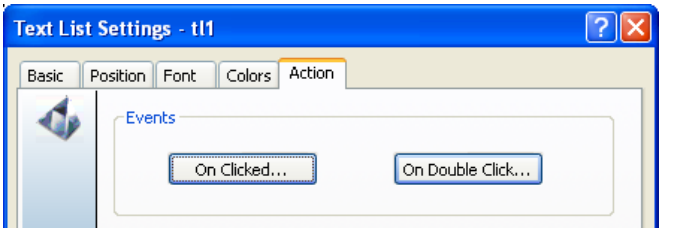
The Background color and the Foreground (Text) color on the Text Input can be specified independently. The enabled colors are used when the element is enabled. Click on the Change… button to bring up the Color screen. You can select from the basic colors on the left, select a color from the color matrix on the right, specify Hue/Saturation/Luminosity, or specify Red, Blue, and Green values to specify a custom color.

Disabled Colors

The Background color and the Foreground (Text) color on the buttons can be specified independently. The disabled colors are used when the button is disabled and not able to be clicked. Some terminal operating system will override the disabled colors specified here.

Action Tab

The Action tab is used to control the elements response to Events.



Events

The Text element has several special event-driven scripts that allow for customized behavior.

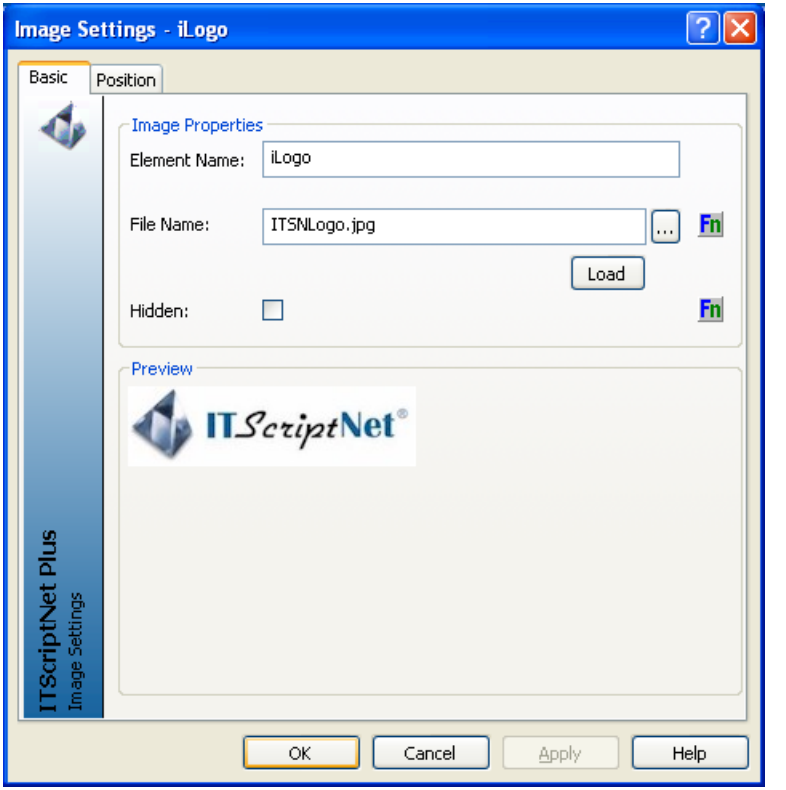
- The **OnClick** Script runs whenever the element is tapped or clicked.
- The **OnDoubleClick** Script runs whenever the element is double-tapped or double-clicked.

Image Element

The Image element is an element for displaying graphics to a user. Use of images can enhance the user interface and improve the look-and-feel of data collection programs.

To add an Input Image Element to a prompt, click on the Image element 

from the element list on the right side of the main Generator Application screen. If you cannot see the element list, you can turn it on by checking the Elements menu under the View main menu item. After you click the Image Element, the Image Settings screen will be displayed. You will need to specify the settings for your Image Element. The key settings are the element Name and the filename of the graphic you wish to display. These and all Image Settings are discussed below.



Basic Tab

The Basic Tab on the Image Settings screen is shown here.

Element Name

Image Elements, like all elements, must have a name. When you first create the Image Element the name will be displayed as “Image1”. You may change the element name to another valid element name if you wish. The name of the element can be up to 20 characters in length and must be a unique name within the prompt. The following characters may not be used in element names: # $ @ _ - * . / = < > ( ) & as they are reserved.

File Name

The File Name is the name of the graphic file for the image to display. Use the Browse button  to locate the graphic file for the Image Element. **IT*Script*Net** supports bitmap (.bmp), jpeg (.jpg), and .png image file formats. The full path to the graphic file must be specified. If there is no path, as shown in the example, the graphic file must be located in the same directory as the data collection program file (.itb file). The image file used for the Image element will be automatically added to the Support Files list. The support files are used by the data collection program and

would typically be sent to the terminal with the data collection program. Please refer to the section in the User Guide on Support Files for more information.

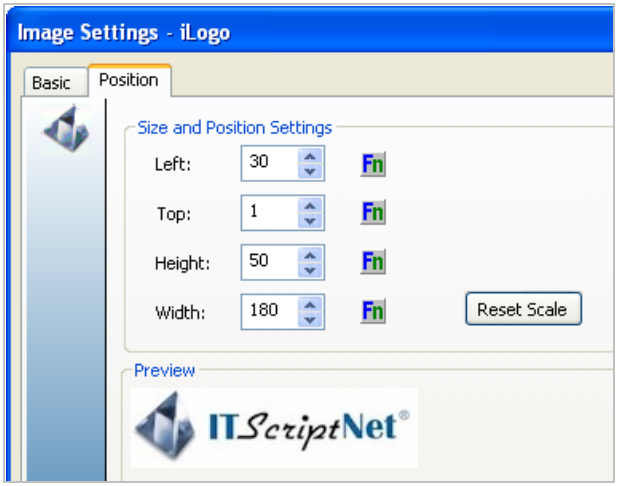
Load Button

The Load button will read the graphic file and refresh the preview of the image that is displayed. You would use the Load button if you type in a file name instead of browsing for the file or if you modify the graphic file and want **IT*Script*Net** to reload it to have the current image.

Hidden

The Hidden setting will cause the Image element to be hidden when checked.

Position Tab



The Position tab on the Image element screen controls the size and position of the Image element on the prompt. You can also control the size and position of the Image element from the main design area. You can move the Image element by selecting it and holding the mouse while you move the Input Image element to the desired location. You can also resize the Image element by selecting it and dragging one of the resize rectangles.

Left Position

The Left Position setting specifies the horizontal location in pixels of the upper-left corner of the Image element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Left Position setting is set to 0, the Image element will be all the way to the left of the prompt.

Top Position

The Top Position setting specifies the vertical location in pixels of the upper-left corner of the Image element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Top Position setting is set to 0, the Image element will be all the way to the top of the prompt.

Height & Width

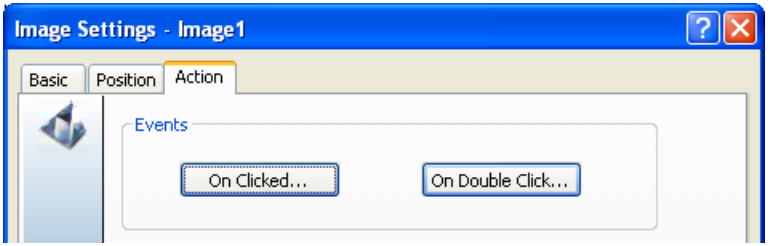
The Height setting specifies the height in pixels of the Image element and the Width setting specifies the width in pixels of the Image element. Note that if you change the height and the width using these settings you may distort the image from its original state.

Reset Scale

The Reset Scale button will reset the height and width of the image according to the original height and width of the image as stored in the graphics file.

Action Tab

The Action tab is used to control the elements response to Events.



Events

The Text element has several special event-driven scripts that allow for customized behavior.

- The **OnClick** Script runs whenever the element is tapped or clicked.
- The **OnDoubleClick** Script runs whenever the element is double-tapped or double-clicked.

Shape Element

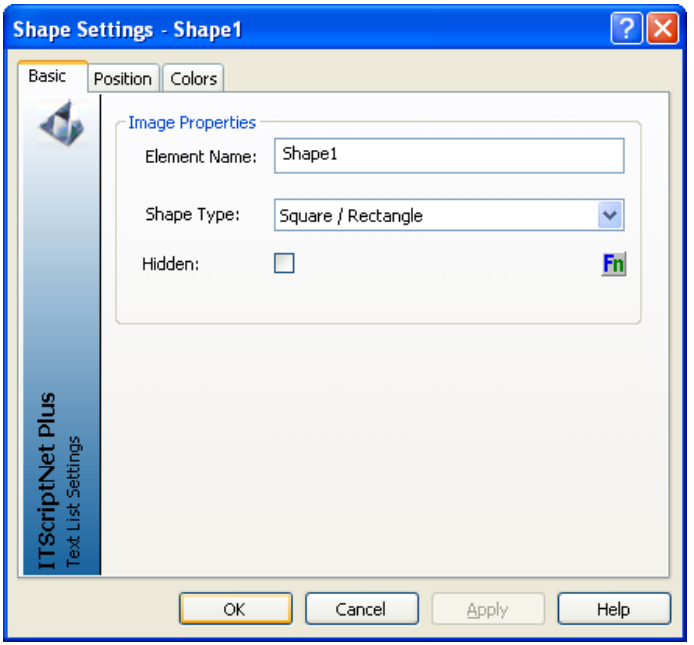
The Shape element is an element for displaying an enhanced user interface in the data collection programs.

To add a Shape element to a prompt, click on the Shape element 

from the element list on the right side of the main Generator Application screen. If you cannot see the element list, you can turn it on by checking the Elements menu under the View main menu item. After you click the Shape element, the Shape Settings screen will be displayed. You will need to specify the settings for your Shape element. The key settings are the Element Name and the type of shape.

Basic Tab

The Basic tab on the Shape Settings screen is shown here.



Element Name

Shape elements, like all elements, must have a name. When you first create the Shape element the name will be displayed as “Shape1”. You may change the element name to another valid element name. The name of the element can be up to 20 characters in length and must be a unique name within the prompt. The following characters may not be used in element names: # $ @ _ - * . / = < > ( ) & as they are reserved.

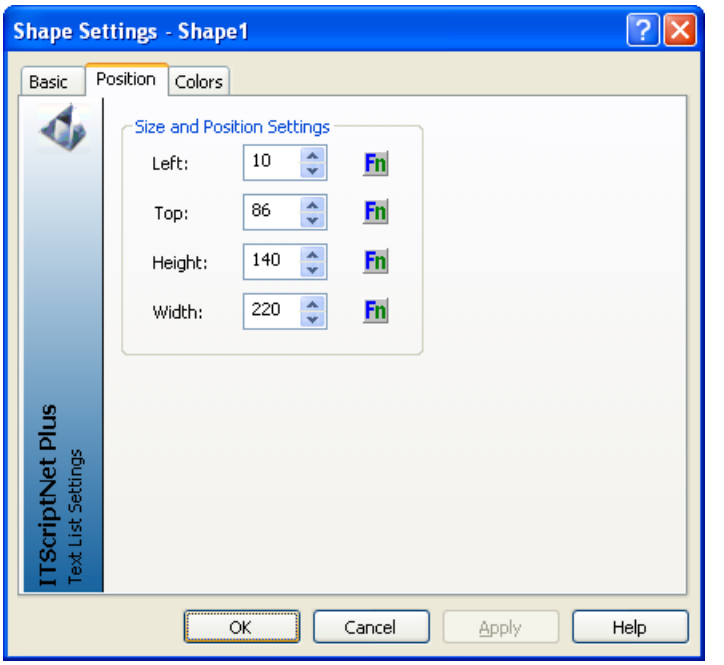
Shape Type

This is the type of shape to include on the prompt. The three choices for shape type are Square/Rectangle, Circle/Oval, or Line.

Hidden

The Hidden setting will cause the Shape element to be hidden when checked.

Position Tab



The Position tab on the Shape element screen controls the size and position of the Shape element on the prompt. You can also control the size and position of the Shape element from the main design area. You can move the Shape element by selecting it and holding the mouse while you move the Shape element to the desired location. You can also resize the Shape element by selecting it and hovering over the resize boundaries of the Shape element and dragging to resize it.

Left Position

The Left Position setting specifies the horizontal location in pixels of the upper-left corner of the Shape element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Left Position setting is set to 0, the Shape element will be all the way to the left of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the left position to a value larger than the screen the element will be located off the prompt screen.

Top Position

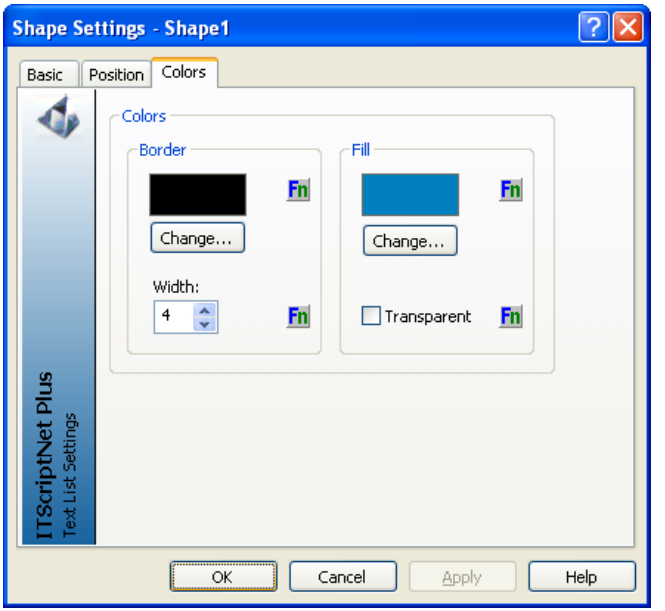
The Top Position setting specifies the vertical location in pixels of the upper-left corner of the Shape element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Top Position setting is set to 0, the Shape element will be all the way to the top of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the top position to a value larger than the screen the element will be located off the prompt screen.

Height & Width

The Height setting specifies the height in pixels of the Shape element and the Width setting specifies the width in pixels of the Shape element.

Colors Tab

The Colors tab adds another dimension of flexibility and customization to the design of Shape elements. The colors will only be effective for portable terminals that have a color display.



Border Color

The border color can be set by clicking the Change… button to bring up the Color screen.

Border Width

This setting controls how many pixels wide the border will be around the shape.

Fill Color

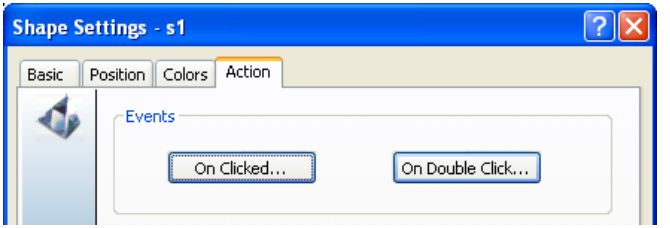
The fill color for the shape is separate from the border color. Click the Change… button to control the fill color of the shape.

Transparent

The Transparent setting will cause the shape to be filled as transparent instead of with a color. If you specify a color and then check the transparent box, the fill color will be ignored and the transparent shape will allow elements beneath it to show through.

Action Tab

The Action tab is used to control the elements response to Events.



Events

The Text element has several special event-driven scripts that allow for customized behavior.

- The **OnClick** Script runs whenever the element is tapped or clicked.
- The **OnDoubleClick** Script runs whenever the element is double-tapped or double-clicked.

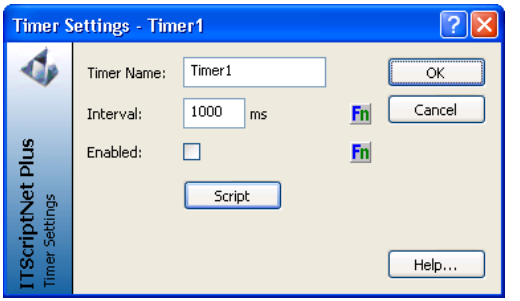
Timer Element

This element is used to execute a script on a time interval. You can enable and disable the timer, or allow it to run and execute the script every time the interval expires.

To add a Timer element to a prompt, click on the Text element 

from the element list on the right side of the main Generator Application screen. If you cannot see the element list, you can turn it on by checking the Elements menu under the View main menu item. After you click the Timer element, the Timer Settings screen will be displayed. You will need to specify the settings for your Timer element. The key settings are the element Name and the Interval. These and all Timer Settings are discussed below.

Timer Name



Timer elements, like all elements, must have a name. When you first create the Timer element the name will be displayed as “Timer 1”. You may change the element name to another valid element name if you wish. The name of the element can be up to 20 characters in length and must be a unique name within the prompt. The following characters may not be used in element names: # $ @ _ - * . / = < > ( ) & as they are reserved.

Interval

This is the time interval to run the script. This value is in milliseconds.

Enabled

This property determines whether the timer is running or not. If the timer is enabled, the script will be run each time the interval expires.

Script

This button brings up the script editor, used to enter the script for the timer.

Enabling and Disabling the Timer

You can use the *Enable* and *Disable* functions to control the timer. If the timer is enabled, it will execute the script once every time the time interval is reached. If you want the timer to only run once, you would call *Disable* in the script for the timer.

Using the Timer

Referencing the element with the $prompt.element$ syntax returns the interval. Setting the interval by assigning a value to $prompt.element$ stops and restarts the timer with the new value.

Button Element

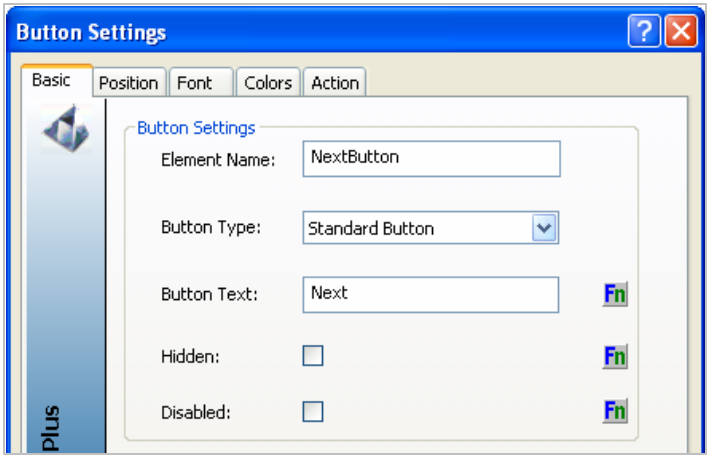
Buttons are a key element type for Multi-Prompts, and are a unique type of element for program design. They are considered to be Input elements because users will click on the buttons to make actions occur, but there is no data collected for buttons like the other true input elements. You will almost always have at least one button on your prompts. When a button is clicked, the button will either accept the prompt's input and move to the next prompt, escape to the escape prompt, or perform some other action that is defined in the button's OnClicked script.

To add a button to a prompt, click on the button element 

from the element list on the right side of the main Generator Application screen. If you cannot see the element list you can turn it on by checking the Elements menu under the View main menu item. After you click the button element the Button Settings screen will be displayed. You will need to specify the settings for your button. The key settings are the Element Name, the Button Text, and the Button Press Action. These and all Button Settings are discussed below.

Basic Tab

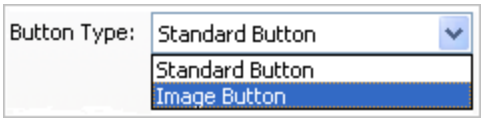
The Basic Tab on the Button Settings screen is shown here.



Element Name

When you first create the button the name will be displayed as "Button1", but you may change it to any valid element name. Element names can be up to 20 characters long and must be unique within the prompt. The following characters may not be used on prompt names: # $ @ _ - * . / = < > ( ) & as they are reserved.

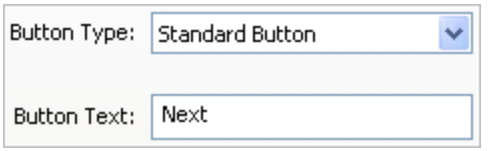
Button Type



A button can be either a Standard Button or an Image Button. The only difference between a standard button and an image button is that a standard button will display text and an image button will display the button with an image. Select the type of button that you want in the drop-down.

Button Text

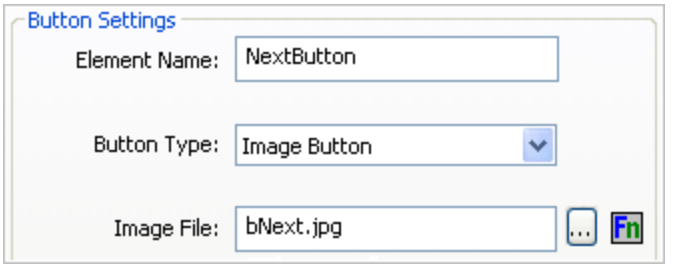
The Button Text setting is only applicable to Standard Buttons. The Button Text is the text that is displayed on the button. In the example shown, the Button Text



setting is filled in with “Next”. The resulting button will look like this: 

Image File

The Image File setting is only applicable to Image Buttons. The Image File is the filename of the graphic to be displayed on the button. The Browse button (…) is available to locate graphics files (.bmp, .jpg, .png) to use for the image buttons. In the example shown, the image file used is “bNext.jpg”. The resulting buttons will look like this:



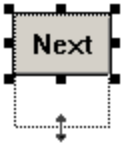
Using image buttons can enhance the look of the data collection program and make it more appealing and intuitive for the data collection user.

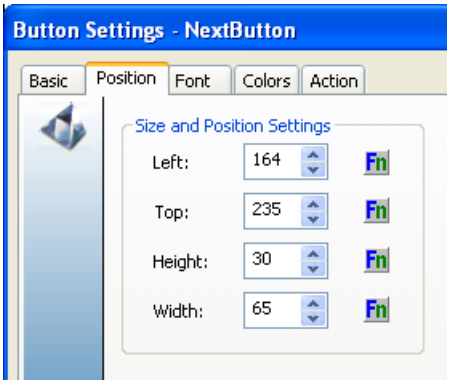
Hidden, Disabled

The Hidden Setting will cause the button to be hidden when checked. The Disabled Setting will cause the button to be disabled. A button that is hidden and/or disabled cannot be clicked.

Position Tab

The Position Tab on the Button Settings screen controls the size and position of the button element on the prompt. You can also control the size and position of the button element from the main design window. You can move the button element by selecting it and holding the mouse while you move the button to the desired location. You can also resize the button element by selecting it and dragging the resize rectangles.





Left Position

The Left Position Setting specifies the horizontal location in pixels of the upper-left corner of the Button Element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Left Position Setting is set to 0, the button element will be all the way to the left of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the left position to a value larger than the screen the element will be located off the prompt screen.

Top Position

The Top Position Setting specifies the vertical location in pixels of the upper-left corner of the Button Element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Top Position Setting is set to 0, the button element will be all the way to the top of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the top position to a value larger than the screen the element will be located off the prompt screen.

Height

The Height Setting specifies the height in pixels of the Button.

Width

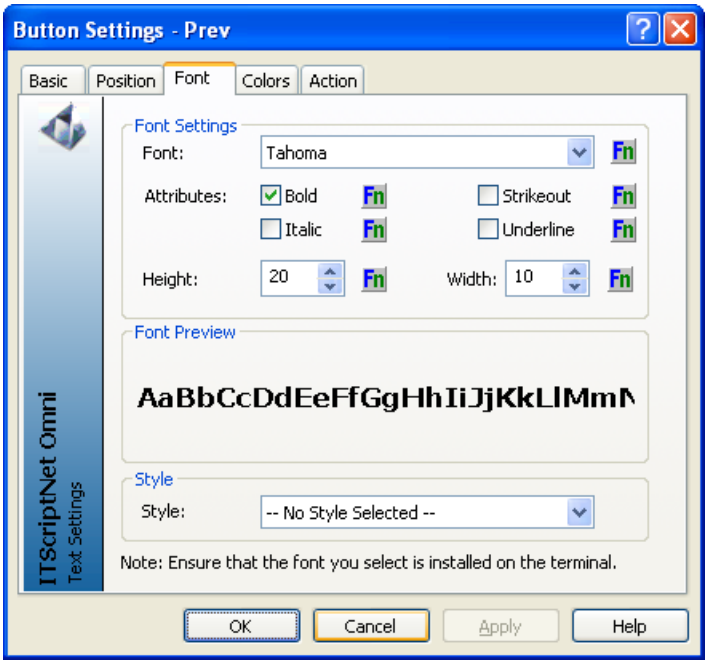
The Width Setting specifies the width in pixels of the Button.

Font Tab

The Font Tab on the Button Settings screen allows you to customize standard buttons. The font settings will have no effect on image buttons.

Font

The Font selection allows you to choose a font for the button's text. Note that if you select a font that is not installed on your portable terminal, the operating system of the terminal will convert the font to a font that is installed. The effect of this conversion will most likely be that the positioning of the font on the button on the terminal will not be consistent with the design view. The area under the font selection displays a preview of the font you have chosen.



Attributes

The Attributes section allows you to include Bold, Italic, Underline, or Strikeout attributes to your font choices. A check next to the attribute turns that attribute on.

Height & Width

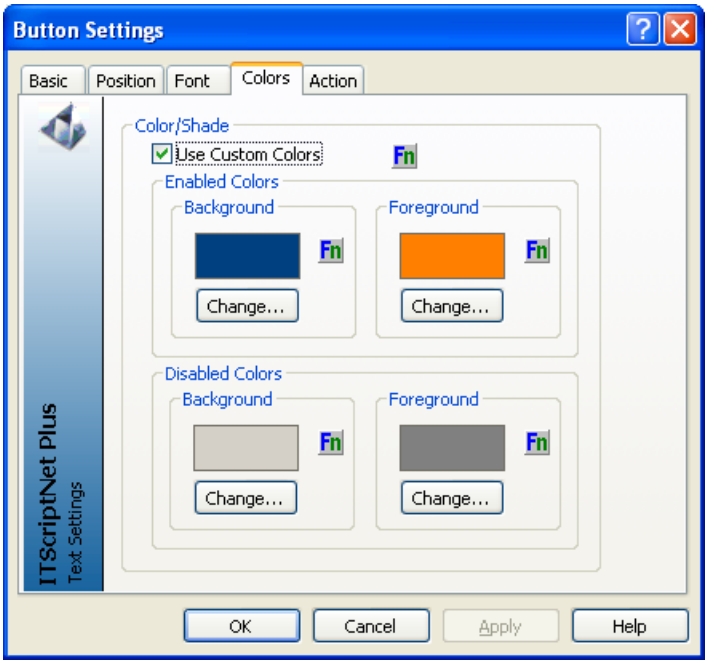
The Height and Width Font Settings specify the Height and Width of the Font size in pixels. The Font Preview section will reflect the Height and Width specified. If you use a Font Width of zero, Windows will select an appropriate default width.

Style

The Style setting allows you to apply a pre-defined Font and Color set to this element. The Style must have been created with the Style Editor on the Program menu. Any changes made to the Style in the Style Editor will automatically be applied to this element. If you change any Font or Color setting on the element, the Style will be removed.

Colors Tab

The Colors Tab on the Button Settings screen adds another dimension of flexibility and customization to the design of data collection prompts and Button Elements. The colors will only be effective for portable terminals that have a color display. The first step towards customizing the colors of a standard button is to check the 'Use Custom Colors' option on the Colors tab. Once checked, the colors specified on the tab will be used. Here is an example of a Standard Button that a user would click to move to the next prompt. This button uses a larger bold and Italic font and custom colors to really add some flair as compared to the typical gray button:



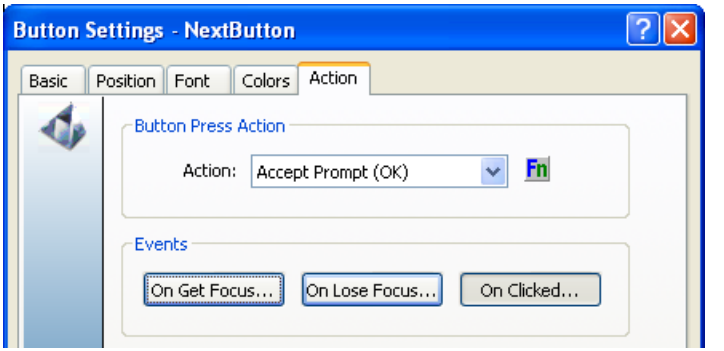
Enabled Colors

The background color and the Foreground (Text) color on the buttons can be specified independently. These colors are used when the button is enabled. Click on the Change… button to bring up the Color screen to specify the color. You can select from the basic colors on the left, select a color from the color matrix on the right, specify Hue/Saturation/Luminosity, or specify Red, Blue, and Green values to specify a custom color.

Disabled Colors

The background color and the Foreground (Text) color on the buttons can be specified independently. These colors are used when the button is disabled. Note that some Window CE and PocketPC devices override the disabled foreground color.

Action Tab



The Action tab contains the Press Button Action setting, which is critical to defining the role of the button on the prompt. This tab also has access to the in-prompt scripts that can be used to customize behavior when the button Gets the Focus, Loses the Focus, or is clicked.

Button Press Action

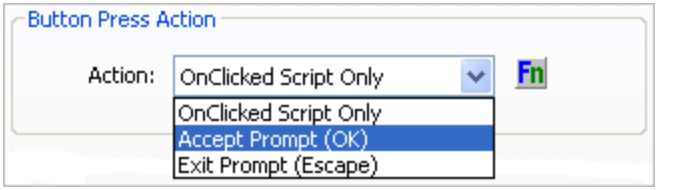
The Action tab for Button Elements contains the Action setting that determines the behavior of the program once a user clicks the button. Select from 3 choices to determine the button's behavior.

On Clicked Script Only

The 'On Clicked Script Only' action will run the On Clicked script (see Events below).

Accept Prompt (OK)

The 'Accept Prompt (OK)' action means that when the user clicks the button, the entire prompt will be accepted and the program will proceed to the Next prompt as defined in the prompt's settings. The On Clicked in-prompt script will still run. There would typically be a button on each Multi-Prompt that would be set to be the Accept button (or Next, OK, etc.) so that the user can proceed to the next prompt for data collection.



Exit Prompt (Escape)

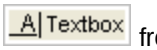
The 'Exit Prompt (Escape)' action will cause the program to abort, or escape, from the prompt and go to the prompt designed as the Escape prompt in the Prompt's settings. There would typically be a button on each Multi-Prompt that would be set to be the Escape button (or Exit, Previous, etc.) so that the user can escape to abort the prompt and often go back to the previous prompt.

Events

The Button Elements have three special event-driven scripts that allow for customized behavior. There is the "On Get Focus" script, the "On Lose Focus" script, and the "On Clicked" script. The On Clicked script is used to specify the behavior of the button when it is clicked. This script must be used if the button is to have any other or additional functionality than just Accept or Escape.

Input Text Element

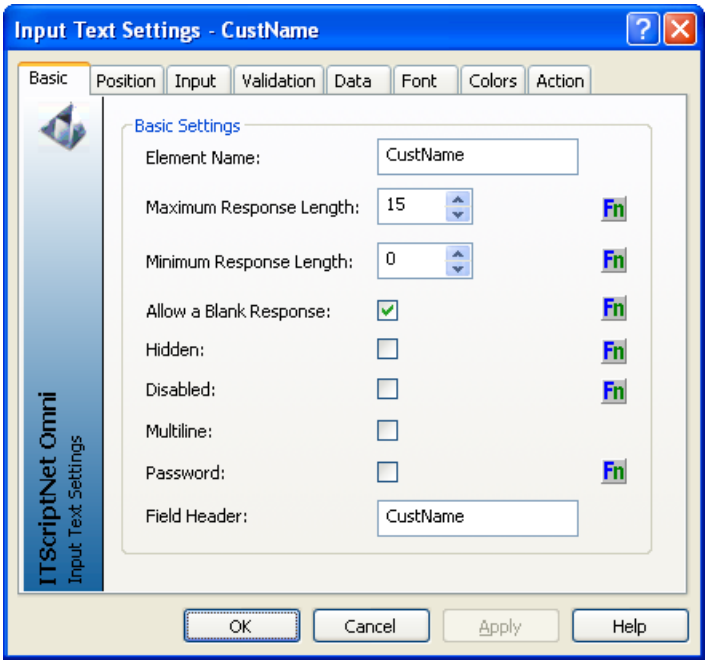
The Input Text element is a Textbox for entering data into the data collection programs. The data can be entered from the keypad of the terminal or scanned.

To add an Input Text element to a prompt, click on the Textbox element 

from the element list on the right side of the main Generator Application screen. If you cannot see the element list, you can turn it on by checking the Elements menu under the View main menu item. After you click the element button, the Input Text Settings screen will be displayed. You will need to specify the settings for your new element. The key settings are the element Name, the minimum and maximum response lengths for Input Text, and whether the input should be entered with the keypad or scanned. These and all Input Text Settings are discussed below.

Basic Tab

The Basic tab on the Input Text Settings screen is shown here.



Element Name

Input Text elements, like all elements, must have a name. When you first create the Input Text the name will be displayed as "TextBox1". You can rename the element to a valid element name. An element name can be up to 20 characters in length and must be a unique name within the prompt. The following characters may not be used on Input Text element names: # $ @ _ - * . / = < > ( ) & as they are reserved.

Maximum and Minimum Response Length

The maximum and minimum response lengths are properties that set limits on the length of the response entered for the Input Text element. The limits of the number of characters for a response are the most basic form of validating data. For example, if each employee has a time clock number that is 3 or 4 digits, then the settings for the Employee clock number can be set to require at least 3 characters/digits and no more than 4. When collecting data a user will not be able to enter a response longer than the maximum response length. If the user enters a response that is less than the minimum response length, the program will issue a warning to that effect, and the user will not be permitted to move to the next prompt until the response for the input text element is within the minimum and maximum response lengths. If the response is scanned, the program will still check

the length of the scanned response and will reject any response that does not meet the response length criteria.

Allow a Blank Response

This setting will determine whether or not to allow a blank response for the Input Text element. By default this setting is turned on, meaning that a response is not required. If the box is checked, thus allowing a blank response, the user collecting the data will be able to go to the next prompt even if the Input Text element has no response and is blank. Allowing a blank response is useful for optional data in a data collection program. As an example, a note field that allows a user to enter special notes would often be blank unless the user had a special circumstance that required him to key in a note.

Hidden, Disabled

The Hidden setting will cause the Input Text element to be hidden when checked. The Disabled setting will cause the Input Text element to be disabled. A response cannot be entered into an Input Text element that is hidden and/or disabled.

Multiline

The Multiline Setting will allow an Input Text element to be created so that its height is taller than one line of input text and will allow the text the user enters (or scans) to wrap to any number of lines (as long as the number of characters does not exceed the maximum response length). Scrolling up and down inside a multiline input text element is accomplished with the arrow keys.

Password

The Password Setting controls how text is displayed in the Input Text element. If this checkbox is set, text will be displayed as '*' in a password mode. Otherwise, the text will be displayed normally.

Database Field/Column Header/Field Header

This setting specifies where the data collected for this element is stored after it is sent to the PC and processed. Depending on the settings specified in the *Configure Receive* screen, this field may have different meanings.

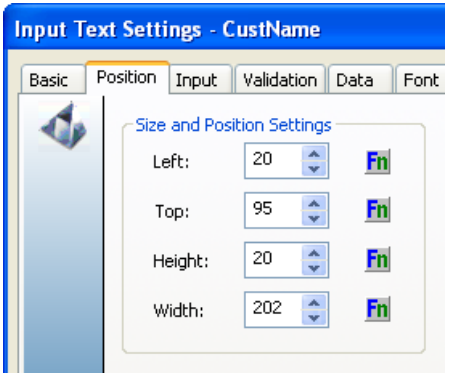
If the program is configured to send its data to a Microsoft Access file or an ODBC data source, the target database field must be set up properly so the data collected can be sent to the desired field. A field from the target database must be selected from the list of fields. The fields in the list are the fields defined for the database and table that are established on the *Configure Receive* screen. The <Ignore> selection can be chosen and the collected data for the Text Input element will be ignored when the collected data is sent to the database. Fields are output to Access and ODBC if the field specified here exists and the maximum length of the response is not zero.

If the program is configured to send its data to a Microsoft Excel file, this setting will become the column header for the data. For Excel files, if this field is left blank, the element name will be used as the column header.

To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen.

If the program is configured to send its data to a text file with a header, this setting will become the field header for the data in the output file. For text files with headers, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen. If the program is configured to send its data to a text file with no header information, then there will be no Field Header setting on the Input Text Settings screen.

Position Tab



The Position Tab on the Input Text element screen controls the size and position of the Input Text element on the prompt. You can also control the size and position of the Input Text element from the main design window. You can move the Input Text element by selecting it and holding the mouse while you move the Input Text element to the desired location. You can also resize the Input Text element by selecting it and hovering over the resize boundaries of the Input Text element and dragging the Input Text element to resize it.

Left Position

The Left Position setting specifies the horizontal location in pixels of the upper-left corner of the Input Text element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Left Position setting is set to 0, the Input Text element will be all the way to the left of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the left position to a value larger than the screen the element will be located off the prompt screen.

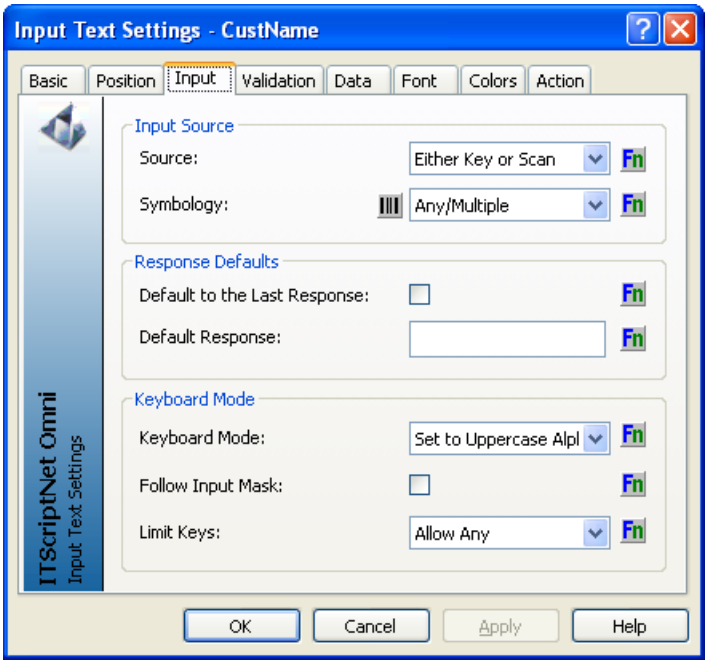
Top Position

The Top Position setting specifies the vertical location in pixels of the upper-left corner of the Input Text element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Top Position setting is set to 0, the Input Text element will be all the way to the top of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the top position to a value larger than the screen the element will be located off the prompt screen.

Height & Width

The Height setting specifies the height in pixels of the Input Text element and the Width setting specifies the width in pixels of the Input Text element.

Input Tab

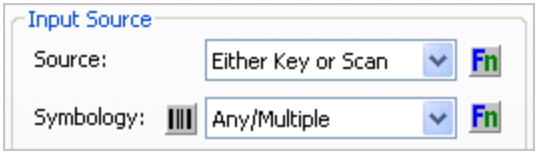


The Input Tab contains the settings that control the response input behavior of the Input Text Element. This tab controls whether the data for the user's response must be entered on the keypad, scanned, or if either is allowed. This is also the tab that contains specific information about bar code symbologies and if the Input Text element is designed to require a certain symbology. Default settings for the response are also on this tab.

Input Source

Source

The setting for source specifies whether the response to the Input Text element can be entered only on the terminal's keypad, only by scanning a barcode, or by either keypad or scan. By default, Input Text elements are set to allow either keypad or scan for data entry. Depending on the application, requiring barcode scans will *reduce data collection time* (scanning is faster than punching in the data on a keypad) and *maximize accuracy*.



Symbology

The setting for Symbology is used to limit the allowed barcode symbology when scanning a response for an Input Text element. There are many different barcode symbologies. A barcode symbology is an algorithm for encoding data into a barcode. Some of the more common symbologies are: Code 39, I2of5, Code 128, and UPCA. The Input Text element can be set to accept any barcode symbology (the terminal has to be able to decode the symbology, please refer to documentation on the specific portable terminal you are using for a list of symbologies compatible with your hardware). The Any/Multiple selection will allow a user to scan any symbology that is configured as a default for the scanner/imager for your hardware. The Any/Multiple setting is the default for any new Input Text elements that you create.

The Symbology property can be set to accept only one type of barcode symbology. To select a single symbology, select it from the list. The default settings

for the symbology on your terminal will be used. By way of example, let's say that your warehouse uses Code 39 barcodes for your part numbers. However, the boxes also contain UPC codes for those products. When collecting data, you want to insure that the Code 39 barcode is scanned and not the UPC code. By setting the Input Text element to only accept Code 39, the program will reject the UPC codes and any other symbology. Specifying a specific symbology is a means of validating your data and *maximizing data accuracy.*

Depending on your data collection needs, you may need to customize the Symbology setting. In our example above, the warehouse had Code 39 and UPC part numbers. Let's say that now you need to be able to scan either a Code 39 or a UPC, but still do not want to allow any other symbologies. You can do this by clicking on the button with the barcode icon to bring up the *Advanced Symbology Settings* screen. Please refer to the section in this User Guide describing the *Advanced Symbology Settings* screen. The *Advanced Symbology Settings* screen will also allow you to control each parameter available for one or more symbologies. If you do not customize the barcode symbology settings, the default settings for the one symbology specified or all symbologies will be used. The defaults are appropriate for most data collection applications, but if you need to control the settings more precisely, **IT*Script*Net** allows you access and full control of the symbologies. If the *Advanced Symbology Settings* screen is used to customize the symbology behavior for an Input Text element, the barcode icon will change colors to provide an easy visual indicator that customized symbology settings are in use.

Response Defaults

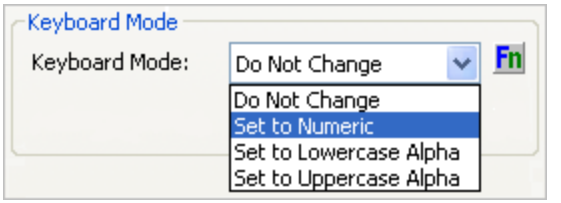
The Response Defaults settings can set the default response for the element to a specific value or the value of the response from the last data record collected for the element. When creating a new input text element, the response default is initially set to no default. When in this state, the user will have to enter data for the element each time through the prompting loop during data collection.

Using response defaults, the *Default to the Last Response* option can be enabled and will cause the response to default to the previous response. The user will still be able to change the response, but can save time by accepting the default. This feature is useful when an element will often have the same information for several prompting loops. For example, an asset physical inventory requires a location. The user would enter a location and then scan all the asset tags within that location before moving on to the next one. With the *Default to Last Response* option enabled, the location need only be entered the first time and then the field will default to that location until the user moves to another location and then enters a new location code.

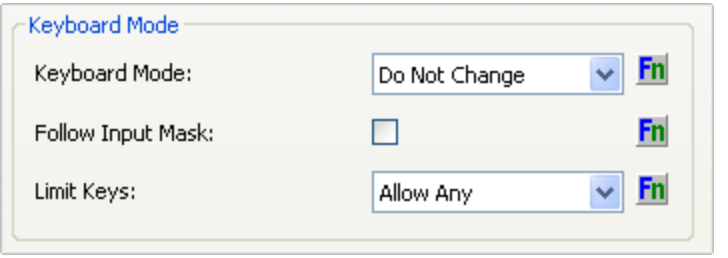
The other option available for the response default is to actually set a response default that will always be shown as the default response for the input text element. The example shows a value of '100' as the default response for this prompt. This might be useful as an employee code, for example, in a situation where employee 100 is primarily responsible for collecting data with the program. Another employee could enter or scan his employee code to override the default, but having the default value saves time during data collection most of the time because the user simply can accept the default by pressing the Enter key.

Keyboard Mode

The *Keyboard Mode* property determines whether or not the keypad mode for the terminal will be set explicitly. If this property is *Set to Numeric*, **IT*Script*Net** will force the portable terminal into numeric data entry mode. Using this setting is especially useful when designing data collection programs for portable terminals that have keys that are combination alpha/numeric. If the Input Text element requires the user to enter a quantity, for example, automatically switching the terminal into numeric key mode will save the user time since the user will not have to switch the keypad into numeric mode. An Input Text element can also be configured to *Set to Uppercase Alpha* or *Set to Lowercase Alpha*, which are useful when the prompt's data is expected to be alphabetic. The default for this property is *Do Not Change*, which means that **IT*Script*Net** does not change the keypad mode for the prompt. The options available in this list will depend on the terminal type selected for the program.

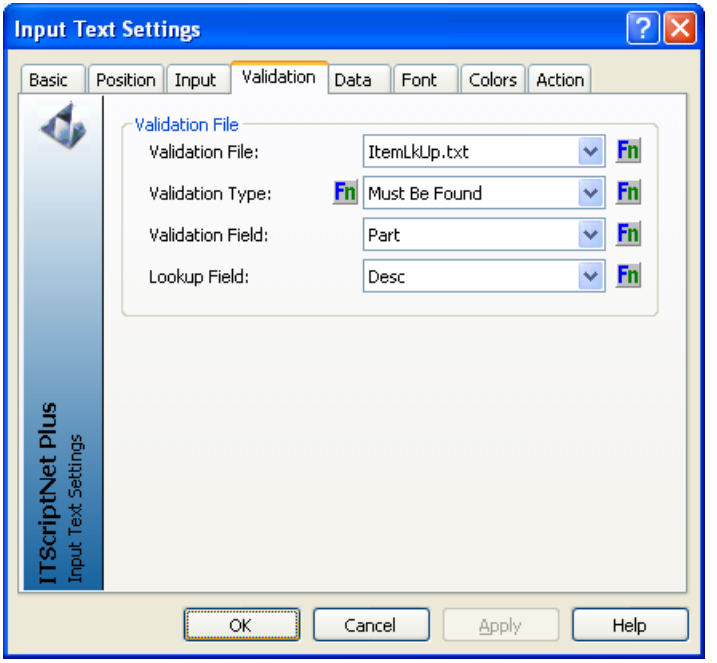


There is also a second option for the keyboard mode. The 'Follow Input Mask' option works in conjunction with the Input Mask setting on the Data tab. When the 'Follow Input Mask' option is checked, the keyboard mode will switch to the appropriate mode to follow the Input Mask. If the Input Mask for the Input Text element is '@##' for example, the Keyboard Mode will first be set to alpha mode since the first character in the mask requires an alpha character. Then the keyboard mode will switch to numeric for the second and third character since the mask requires numeric input for the last two characters for the input.



The last option, 'Limit Keys', allow you to specify that only *Uppercase Letters*, *Lowercase letters*, or *Numbers* can be entered into the field. If one of these selection is used, any other characters typed into the field will be ignored.

Validation Tab



Validation files are text files that can be loaded into the terminal and used to ensure that the operator enters accurate data. When the operator enters a response, the entered data can be compared to the data in the validation file to make sure it is allowed. Additionally, a second field can be retrieved from the validation file and displayed to the operator. Typical uses might include:

- ❑ Part Number/Description lookups and validations
- ❑ Location Validation
- ❑ Employee Number Validation
- ❑ Order Picking
- ❑ Route/Delivery Lists
- ❑ Many More….

The validation text file must be defined from the *Validation Files* screen before a prompt can be configured to use the validation text file. Please refer to the *Validation Files* Screen section of this User Guide for more information on setting up the validation file for your data collection program.

For the purposes of this section, we will assume that a text file named ItemLkUp.txt has been setup for use by our data collection program. The text file contains a part number and a description. The drop-down selection box for the validation file will contain a list of all the validation files that have been identified for use by this program from the *Validation Files* screen. You can select which file you want to use to validate the response for this element. In the example, the ItemLkUp.txt file has been selected. The second validation drop-down box contains the type of validation.

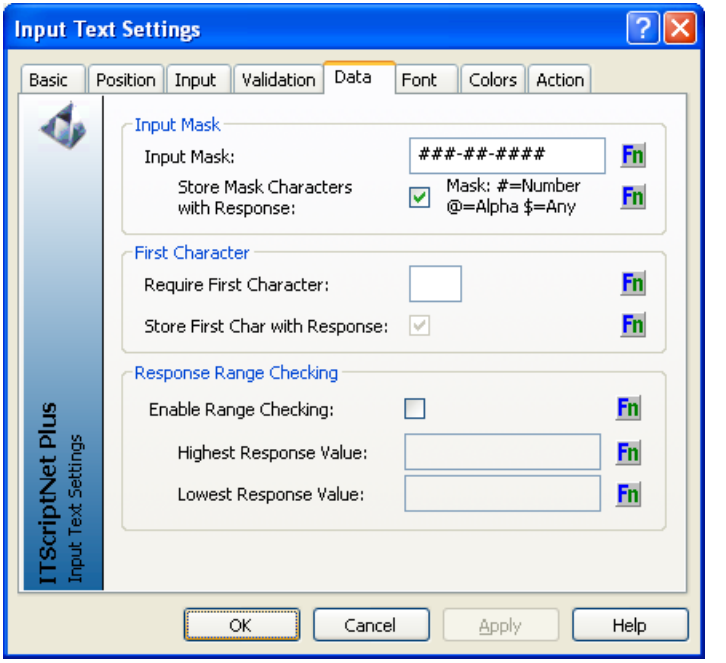
There are three validation modes.

- **Lookup Only:** This is used to perform a lookup, but does not require the response to exist.
- **Must Be Found:** This mode means that the entered data must exist in the lookup file. If the response entered is not found in the validation file, the response is rejected.
- **Must Not Be Found:** This mode means that the entered data must NOT exist in the lookup file. If the response is already in the lookup file the response is rejected. In our example, we have selected the 'Must Be Found' validation since we want to validate the part number entered or scanned is a valid part number.

The third drop-down selection is the field name to be used to validate this Input Text element. When the ItemLkUp.txt file was added to the list of validation files for this program, the definition for that file was also configured. One of the fields in the file was named 'Part' and the description included in the validation was named 'Desc'. In our example, the element that we are validating is the part number so the Part field is selected as the Validation Field. The description is selected in the fourth drop-down selection as the information to lookup as the result of the validation. The description can be displayed to the operator later by using the # marker in the display text for a later prompt. The Lookup data is not stored in the collected data file. Only the entered responses are stored in the collected data file.

Data Tab

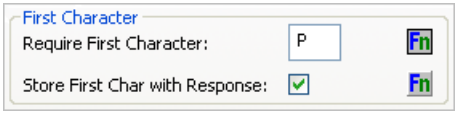
This tab contains three types of built-in rules that can be applied to collected data to insure accuracy of the data you collect.



Input Mask

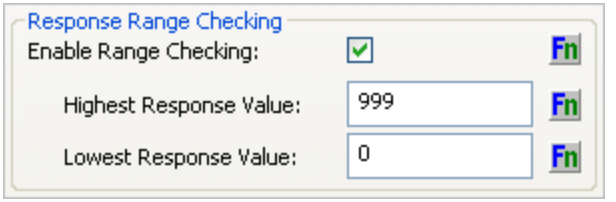
If the Input Mask is used for an Input Text element, the user will be required to enter data conforming to the input mask in order for the data to be accepted as a valid response. The mask is entered as a string of characters. Any required character can be entered as well as any of the three wildcard placeholders in the mask. The # sign requires a number between 0-9 in that location in the input string. A @ symbol requires an alphabetic character (A-Z or a-z) and the $ is a placeholder for any character. In the example shown, the input mask is ###-##-####. This input requires 3 numeric characters, then a dash ('-'), then 2 more numeric characters, another dash ('-'), and then 4 numeric characters to complete the input. This input mask would correspond to a social security number. The input mask by default will store the whole input string, but by unchecking the 'Store Mask Characters with Response' box, the response can be stored without the specific mask characters. In our social security number format example, the number would be stored without the dashes if this option were turned off. **IT*Script*Net** has many data validation options in addition to the input mask. It is therefore important that all the validations be considered when designing the validation scheme for each element so that the validations can work together. If you specify an Input Mask, the Maximum and Minimum Length properties will automatically be set to match the length of the Input Mask and cannot be changed.

First Character



The First Character data validation feature allows the program designer to require the response to the Input Text element to start with the character specified. In the example shown, the letter 'P' is specified as the required first character. If the user scans a barcode or enters data that starts with something other than 'P', the response will be rejected. This feature is perfect for automotive industry labels that require part number barcodes to start with the letter P followed by the part number. The benefit of encoding a 'P' for part number or an 'L' for location (or any other key letter) within a barcode is that the user is protected from scanning the wrong barcode. This type of protection is especially useful if the label contains several different barcodes. The designer of the program has the option of specifying whether or not to store the first letter with the response. In the case of the part number following the letter 'P', it would often be advisable to remove the 'P' prior to storing the part number response so that the collected data can be processed without an extra step to remove the first character 'P'.

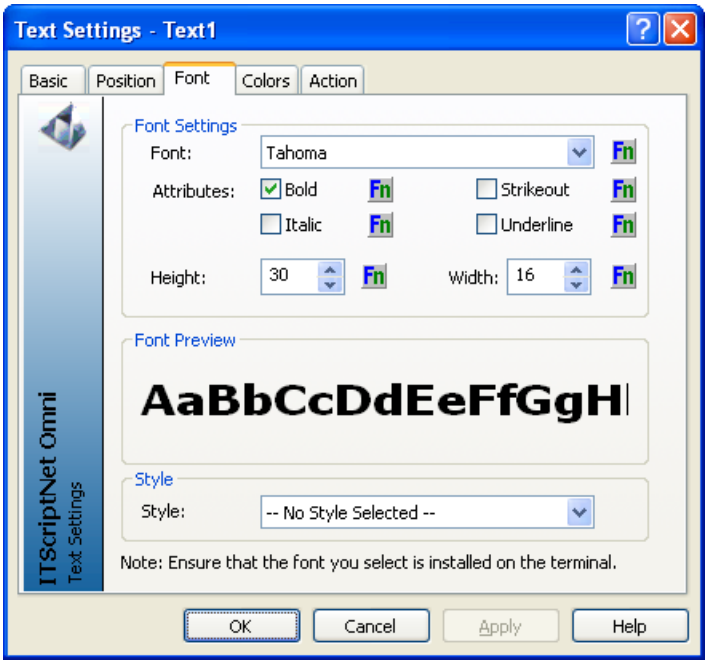
Response Range Checking



The Response Range Checking is another data validation feature of **IT*Script*Net**. When range checking is enabled, the response must be a numeric value. The value of the response will be compared to the high and low response values. If the response entered is within the bounds of the range, the response will be accepted. If the response is higher than the highest response value or lower than the lowest response value set for the range, the response will be rejected and the user will get a message stating that the response was out of range. Enabling the range checking for responses is another way to *increase accuracy* of the data you collect. Response Range Checking works well on elements where numeric data is required, such as quantities.

Font Tab

The Font tab on the Input Text element Settings screen allows you to customize standard Input Text elements.



Font

The Font selection allows you to choose a font for the Input Text. Note that if you select a font that is not installed on your portable terminal, the operating system of the terminal will convert the font to a font that is installed. The effect of this conversion will most likely be that the positioning of the font on the button on the terminal will not be consistent with the design view. The area under the font selection displays a preview of the font you have chosen.

Attributes

The Attributes section allows you to include Bold, Italic, Underline, or Strikeout attributes to your font choices. A check next to the attribute turns that attribute on.

Height & Width

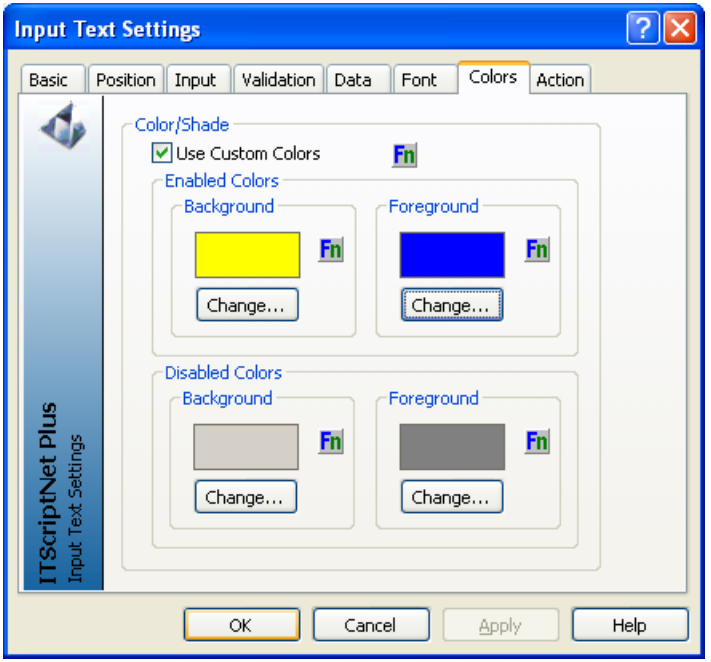
The Height and Width font settings specify the Height and Width of the font size in pixels. The Font Preview section will reflect the Height and Width specified. If you use a Font Width of zero, Windows will select an appropriate default width.

Style

The Style setting allows you to apply a pre-defined Font and Color set to this element. The Style must have been created with the Style Editor on the Program menu. Any changes made to the Style in the Style Editor will automatically be applied to this element. If you change any Font or Color setting on the element, the Style will be removed.

Colors Tab

The Colors tab adds another dimension of flexibility and customization to the design of Text Input elements. The colors will only be effective for portable terminals that have a color display. The first step towards customizing the colors of a standard button is to check the 'Use Custom Colors' option on the Colors tab. Once checked, the colors specified on the tab will be used.



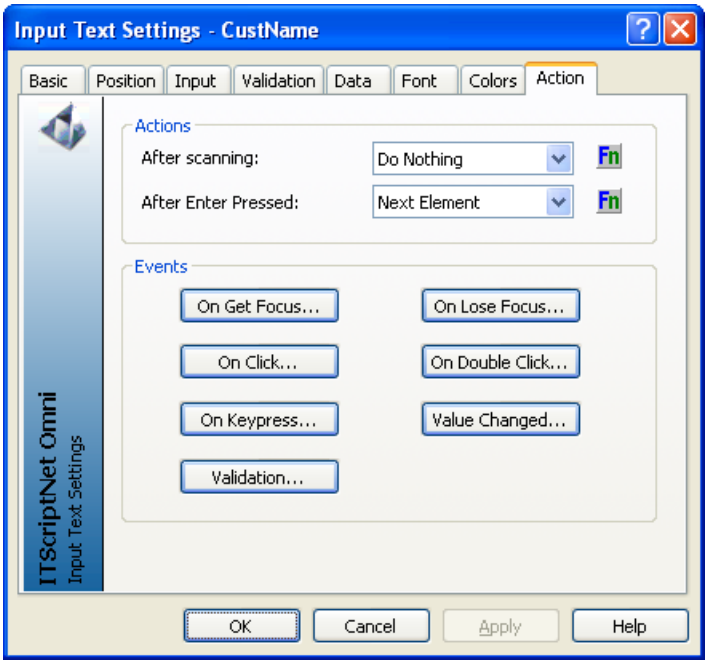
Enabled Colors

The Background color and the Foreground (Text) color on the Text Input can be specified independently. The enabled colors are used when the element is enabled. Click on the Change… button to bring up the Color screen to designate your color choices.

Disabled Colors

The Background color and the Foreground (Text) color on the Input Text element can also be specified for when the Input Text element is disabled. The disabled colors are used when the element is disabled and not able to accept input. Depending on your terminal and terminal's operating system, the disabled colors may be overridden by the operating system of the terminal.

Action Tab

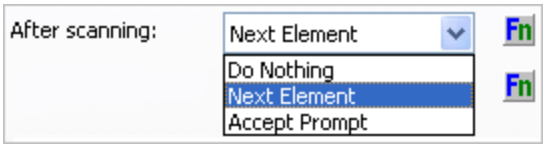


The Action tab contains the settings to control the behavior of the Input Text element after a response has been scanned and/or if the user presses the Enter button. This tab also has access to the in-prompt scripts that can be used to customize behavior when the element gets or loses the focus.

Actions

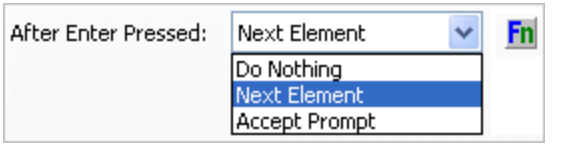
After Scanning

When a user scans a response for the Input Text element (as long as scanning is configured as a valid input source on the Input tab), the After Scanning setting determines how the element will behave. There are 3 choices. The default behavior is for the element to do nothing after the scan. You can choose to have the element advance the focus to the next element on the prompt. The order of the elements is determined on the prompt's settings screen. The last choice is to have the scan act as the acceptance for the entire prompt and thus the program advances to the next prompt as defined on the prompt's settings screen.



After Enter Pressed

When a user enters a response for the Input Text element, it is often desirable to have the data collection program automatically advance to the next element so that the user does not have to click on the next element to keep entering data. To help minimize the number of clicks required for the user, it is often a good idea to take advantage of the After Enter Pressed setting. There are 3 choices for how the element will behave after the Enter button is pressed. The default behavior is for the element to do nothing. You can choose to have the element advance the focus to the next element on the prompt. The order of the elements is determined on the prompt's settings screen. The last choice is to have the Enter key act as the acceptance for the entire prompt and thus the program advances to the next prompt as defined on the prompt's settings screen.



Events

The Input Text element has several special event-driven scripts that allow for customized behavior.

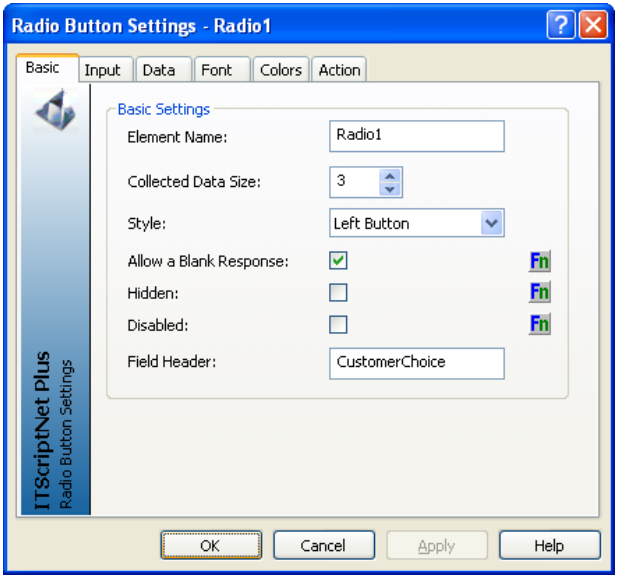
- The **OnGetFocus** script runs when the element receives the focus. This can happen if the user clicks on the element, tabs to the element, if the focus is set to the element in a script, or can occur when a prompt is displayed if the element is the first element on the prompt.
- The **OnLoseFocus** Script runs when the element loses the focus, or in other words, when the focus is moved to another element.
- The **OnClick** Script runs whenever the element is tapped or clicked.
- The **OnDoubleClick** Script runs whenever the element is double-tapped or double-clicked.
- The **OnKeypress** Script runs whenever a key on the keypad is pressed.
- The **ValueChanged** Script runs whenever the data in the control is changed, whether by scanning, data entry, or assigning from a script.
- The **Validation** Script runs when the prompt is accepted, as part of the prompt validation process. If you call the *ValidationFail* function from within this script, the prompt validation will fail, the error message specified will be displayed, and the focus will be set back to this element.

Radio Button Element

The Radio Button element is an element for data collection useful for when you want your user to choose from a set of limited choices. The radio button element is the group of items offering the user his choice. There can be many radio button options within the same Radio Button element.

To add a Radio Button element to a prompt, click on the Radio element 

from the element list on the right side of the main Generator Application screen. If you cannot see the element list, you can turn it on by checking the Elements menu under the View main menu item. After you click the Radio Button element, the Radio Button Settings screen will be displayed. You will need to specify the settings for your Radio Button. The key settings are the element Name, the collected data size, and the data to display and store for the radio button options. These and all Radio Button Settings are discussed below.



Basic Tab

The Basic tab on the Radio Button Settings screen is shown here.

Element Name

Radio Button elements, like all elements, must have a name. When you first create the Radio Button element the name will be displayed as “Radio1”. You may change the name to a valid element name. The element name can be up to 20 characters in length and must be a unique name within the prompt. The following characters may not be used on Input Text element names: # $ @ _ - * . / = < > ( ) & as they are reserved.

Collected Data Size

The collected data size setting is the size of the underlying values that are associated with each choice in the set of radio button items. Each radio button item has a defined display text and a defined value that is stored if that element is selected. The text to display and underlying data values are defined on the Data tab. Please also refer to the section in this User Guide that describes the Data tab of the Radio Button Settings screen.

Style

The Style setting for a Radio Button element determines the relative layout of the radio button circle and the radio button option’s text. The default choice is the Left Button style (shown). The Right Button style puts the circle for the option to the right of the text.

Allow a Blank Response

This setting will determine whether or not to allow a blank response for the Radio Button element. By default this setting is turned on, meaning that a response is not required. If the box is checked, thus allowing a blank response, the user collecting the data will be able to go to the next prompt even if he has not chosen one of the radio button choices and therefore the Radio Button element has no response and is blank. Allowing a blank response is useful for optional data in a data collection program.

Hidden, Disabled

The Hidden setting will cause the Radio Button element and all its items to be hidden when checked. The Disabled setting will cause the Radio Button element and all its items to be disabled. A response cannot be entered, meaning the radio button options cannot be clicked, if the Radio Button element is hidden and/or disabled.

Database Field/Column Header/Field Header

This setting specifies where the data collected for this element is stored after it is sent to the PC and processed. Depending on the settings specified in the *Configure Receive* screen, this field may have different meanings.

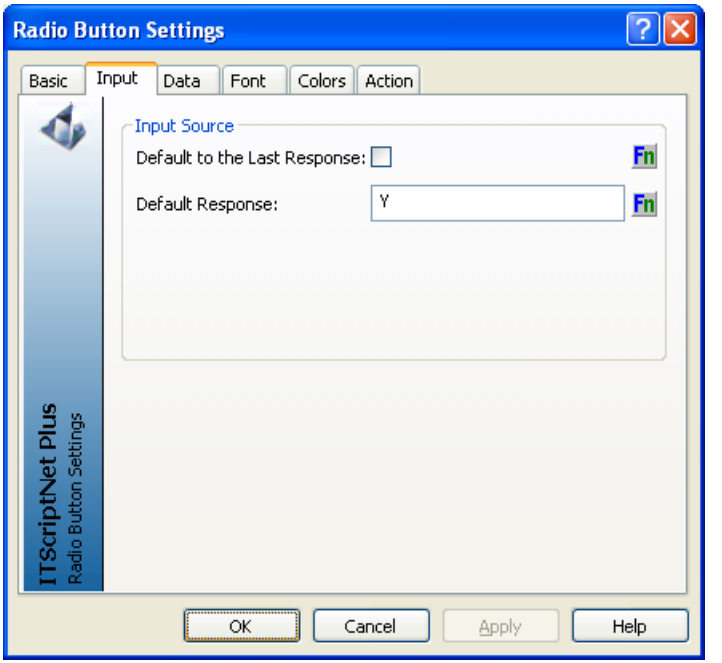
If the program is configured to send its data to a Microsoft Access file or an ODBC data source, the target database field must be set up properly so the data collected can be sent to the desired field. A field from the target database must be selected from the list of fields. The fields in the list are the fields defined for the database and table that are established on the *Configure Receive* screen. The <Ignore> selection can be chosen and the collected data for the Text Input element will be ignored when the collected data is sent to the database. Fields are output to Access and ODBC if the field specified here exists and the maximum length of the response is not zero.

If the program is configured to send its data to a Microsoft Excel file, this setting will become the column header for the data. For Excel files, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen.

If the program is configured to send its data to a text file with a header, this setting will become the field header for the data in the output file. For text files with headers, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen. If the program is configured to send its data to a text file with no header information, then there will be no Field Header setting on the Radio Button Settings screen.

Input Tab

Response Defaults

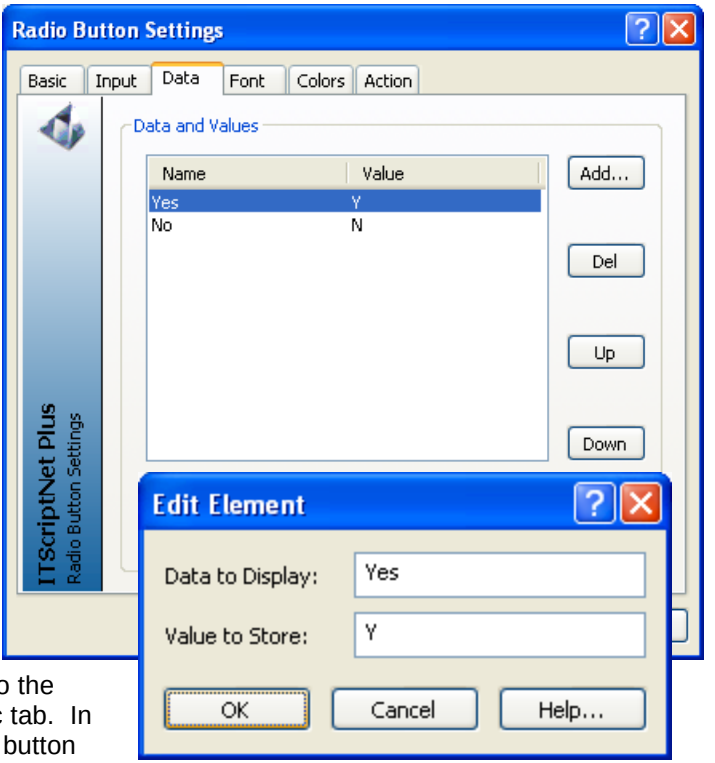


The Response Defaults settings can set the default response for the element to a specific value or the value of the response from the last data record collected. When creating a new Radio Button element, the response default is initially set to no default. When in this state, the user will have to enter data for the element each time through the prompting loop during data collection. For radio buttons, this means that no value will be set. However, a Radio Button element that has the initial focus on a prompt will show one of the radio buttons as selected since a radio button with the focus must have one of the choices selected. It is therefore recommended that a value be explicitly set as the default by an in-prompt script in order to avoid having the program select a radio button option. The Default Response must correspond to one of the radio button item's Value to Store setting as defined on the Data tab.

Using response defaults, the *Default to the Last Response* option can be enabled and will cause the response to default to the previous response. The user will still be able to change the response, but can save time by accepting the default. This feature is useful when an element has the same information for several prompting loops.

The other option available for the response default is to actually set a response default that will always be shown as the default response for the element. The example shows a value of 'Y' as the default response for this Radio Button element that offers a Yes/No choice. This might be useful if the user predominantly will be answering 'Yes' and will only have to click the No option fairly infrequently.

Data Tab



This tab is where you specify the radio button items. The text that is displayed to the user for each radio button item (choice) is defined here as well as the underlying value of the radio button items that is stored as collected data corresponding to the Radio Button element.

Add

Click the Add… button to bring up the Edit Element box. For each radio button choice, you must define the data to display for that radio button item and also the value to store if that choice is selected. As you add radio button items, they will be added to the list that is displayed on the Data tab. Double-clicking an item in the list will bring up the Edit Element screen for editing. The Value to Store for each item should correspond to the Collected Data Size setting on the Basic tab. In the example shown, there are two radio button choices for this Radio Button element. There is a Yes radio button item and a No radio button item. If the user selects the Yes option, the value 'Y' will be stored for this radio button element. If the user selects the No option, the value 'N' will be stored. Shown here also is the resulting Yes/No Radio button with additional display text instructing the user to 'Please Choose One:'

Del

The Delete button removes the selected radio button item from the list.

Up, Down

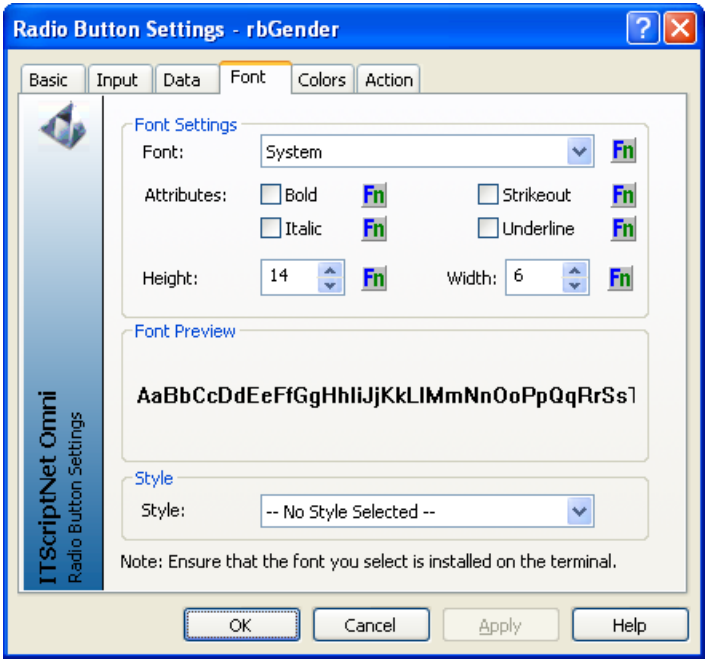
The Up and Down buttons allow you to manipulate the order of the list of radio button choices listed. The Up and Down buttons move the selected radio button item either Up or Down.

Font Tab

The Font tab on the Radio Button Settings screen allows you to customize standard Radio Button element.

Font

The Font selection allows you to choose a font for the Radio Buttons. Note that if you select a font that is not installed on your portable terminal, the operating system of the terminal will convert the font to a font that is installed. The effect of this conversion will most likely be that the positioning of the font for the radio buttons on the terminal will not be consistent with the design view. The area under the font selection displays a preview of the font you have chosen.



Attributes

The Attributes section allows you to include Bold, Italic, Underline, or Strikeout attributes to you font choices. A check next to the attribute turns that attribute on.

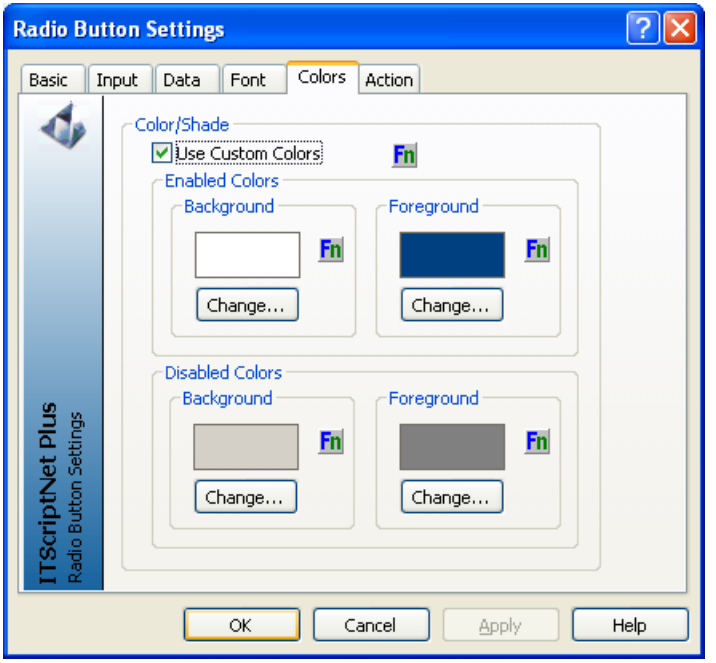
Height & Width

The Height and Width font settings specify the Height and Width of the font size in pixels. The Font Preview section will reflect the Height and Width specified. If you use a Font Width of zero, Windows will select an appropriate default width.

Style

The Style setting allows you to apply a pre-defined Font and Color set to this element. The Style must have been created with the Style Editor on the Program menu. Any changes made to the Style in the Style Editor will automatically be applied to this element. If you change any Font or Color setting on the element, the Style will be removed.

Colors Tab



The Colors adds another dimension of flexibility and customization to the design of Radio Button elements. The colors will only be effective for portable terminals that have a color display. The first step towards customizing the colors of a standard button is to check the 'Use Custom Colors' option on the Colors tab. Once checked, the colors specified on the tab will be used.

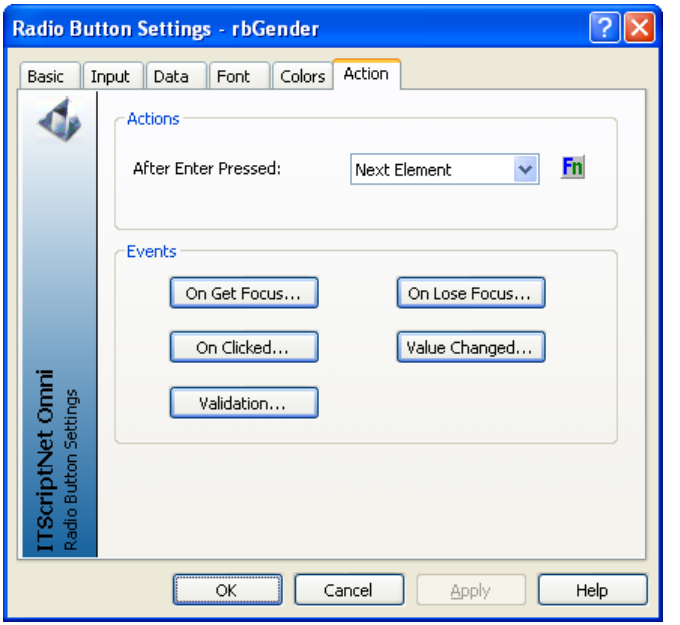
Enabled Colors

The Background color and the Foreground (Text) color on the Text Input can be specified independently. The enabled colors are used when the element is enabled. Click on the Change… button to bring up the Color screen. You can select from the basic colors on the left, select a color from the color matrix on the right, specify Hue/Saturation/Luminosity, or specify Red, Blue, and Green values to specify a custom color.

Disabled Colors

The Background color and the Foreground (Text) color on the buttons can be specified independently. The disabled colors are used when the button is disabled and not able to be clicked. Depending on the terminal, the disabled colors may be overridden by the operating system of the terminal and the disabled colors may not be displayed as designed.

Action



The Action tab contains the settings to control the behavior of the Radio Button element after a response has been registered and if the user presses the Enter button. This tab also has access to the in-prompt scripts that can be used to customize behavior when the elements gets or loses the focus or when any of the radio button options of the Radio Button element get clicked.

After Enter Pressed

When a user enters a response for the Radio Button element, it is often desirable to have the data collection program automatically advance to the next element so that the user does not have to click on the next element to keep entering data. To help minimize the number of clicks required for the user, it is often a good idea to take advantage of the After Enter Pressed setting. There are 3 choices for how the element will behave after the Enter button is pressed. The default behavior is for the element to do nothing. You can choose to have the element advance the focus to the next element on the prompt. The order of the elements is determined on the prompt's settings screen. The last choice is to have the Enter key act as the acceptance for the entire prompt and thus the program advances to the next prompt as defined on the prompt's settings screen.

Events

The Radio Button element has several special event-driven scripts that allow for customized behavior.

- The **OnGetFocus** script runs when the element receives the focus. This can happen if the user clicks on the element, tabs to the element, if the focus is set to the element in a script, or can occur when a prompt is displayed if the element is the first element on the prompt.
- The **OnLoseFocus** Script runs when the element loses the focus, or in other words, when the focus is moved to another element.
- The **OnClick** Script runs whenever the element is tapped or clicked.
- The **ValueChanged** Script runs whenever the data in the control is changed, whether by data entry or assigning from a script.
- The **Validation** Script runs when the prompt is accepted, as part of the prompt validation process. If you call the *ValidationFail* function from within this script, the prompt validation will fail, the error message specified will be displayed, and the focus will be set back to this element.

Checkbox Element

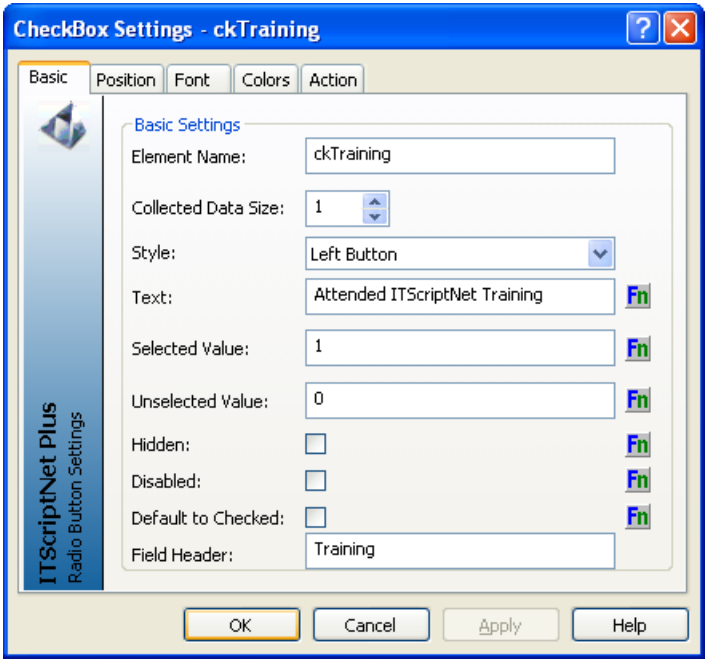
The Checkbox is a data collection element designed to allow a user to select from one of two choices: checked or not checked. The value stored for a checkbox depends on whether the checkbox was selected (checked) or not selected (unselected, not checked).

To add a Checkbox element to a prompt, click on the Checkbox element 

from the element list on the right side of the main Generator Application screen. If you cannot see the element list you can turn it on by checking the Elements menu under the View main menu item. After you click the Checkbox element the Checkbox Settings screen will be displayed. You will need to specify the settings for your Checkbox. The key settings are the element Name, the collected data size, and the data value for the selected and unselected states of the checkbox. These and all Checkbox Settings are discussed below.

Basic Tab

The Basic Tab on the Checkbox Settings screen is shown here.



Element Name

Checkbox elements, like all elements, must have a name. When you first create the Checkbox element the name will be "Checkbox1". You may change the element name to a valid element name. The name of the Checkbox element can be up to 20 characters in length and must be a unique name within the prompt. The following characters may not be used on Input Text element names: # $ @ _ - * . / = < > ( ) & as they are reserved.

Collected Data Size

The Collected Data Size setting is the size of the underlying values that are associated with the selected and unselected state of the checkbox. The data value to store is independent from the text on the prompt displayed to the user. The Collected Data Size should correspond to the Selected Value Setting and the Unselected Value Setting. In the example shown, the Collected Data Size is 1 character, which corresponds to the number of characters in both the Selected and Unselected Value Settings.

Style

The Style setting for a Checkbox element determines the relative layout of the Checkbox square and the Checkbox's text. The Right Button style puts the square for the option to the right of the text. The default choice is the Left Button style.

Text

The text for a checkbox is the descriptive text next to the checkbox square. The example here shows a Checkbox element with the Text set to "Check if option is desired". The Checkbox is a Left Button Style Checkbox and also uses custom fonts and colors described in the sections for the Font and Colors tabs.

Selected Value

The Selected Value is the data value that will be stored for the Checkbox element if the Checkbox is selected (checked).

Unselected Value

The Unselected Value is the data value that will be stored for the Checkbox element if the Checkbox is not selected (not checked).

Hidden, Disabled

The Hidden Setting, when selected, will cause the Checkbox element to be hidden. The Disabled Setting will cause the Checkbox element to be disabled. A response cannot be entered, meaning the Checkbox cannot be clicked, if the Checkbox element is hidden and/or disabled.

Default to Checked

The normal default state for the checkbox is not selected (not checked). If the default for the Checkbox element instead should be checked (selected) then the Default to Checked setting should be turned on.

Database Field/Column Header/Field Header

This setting specifies where the data collected for this element is stored after it is sent to the PC and processed. Depending on the settings specified in the *Configure Receive* screen, this field may have different meanings.

If the program is configured to send its data to a Microsoft Access file or an ODBC data source, the target database field must be set up properly so the data collected can be sent to the desired field. A field from the target database must be selected from the list of fields. The fields in the list are the fields defined for the database and table that are established on the *Configure Receive* screen. The <Ignore> selection can be chosen and the collected data for the Checkbox element will be ignored when the collected data is sent to the database. Fields are output to Access and ODBC if the field specified here exists and the maximum length of the response is not zero.

If the program is configured to send its data to a Microsoft Excel file, this setting will become the column header for the data. For Excel files, if this field is left blank, the element name will be used as the column header. To skip this field and not output it (as with the Ignore selection for Access and ODBC), you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen.

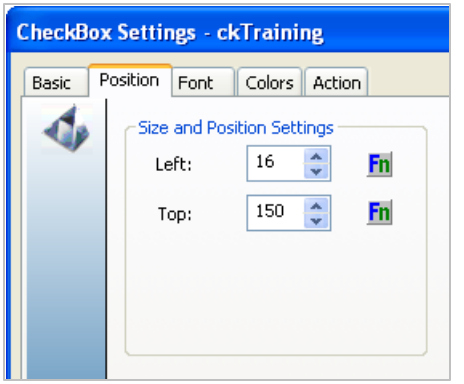
If the program is configured to send its data to a text file with a header, this setting will become the field header for the data in the output file. For text files with headers, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen. If the program is configured to send its data to a text file with no header information, then there will be no Field Header setting on the Checkbox Settings screen.

Position Tab

The Position Tab on the Checkbox element screen controls the position of the Checkbox element on the prompt. You can also control the size and position of the Checkbox element from the main design window. You can move the Checkbox by selecting it and holding the mouse while you move the Checkbox to the desired location. You can also resize the Checkbox by selecting it and hovering over its resize boundaries and dragging the element to resize it.

Left Position

The Left Position Setting specifies the horizontal location in pixels of the upper-left corner of the Checkbox. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Left Position Setting is set to 0, the Checkbox will be all the way to the left of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the left position to a value larger than the screen the element will be located off the prompt screen.

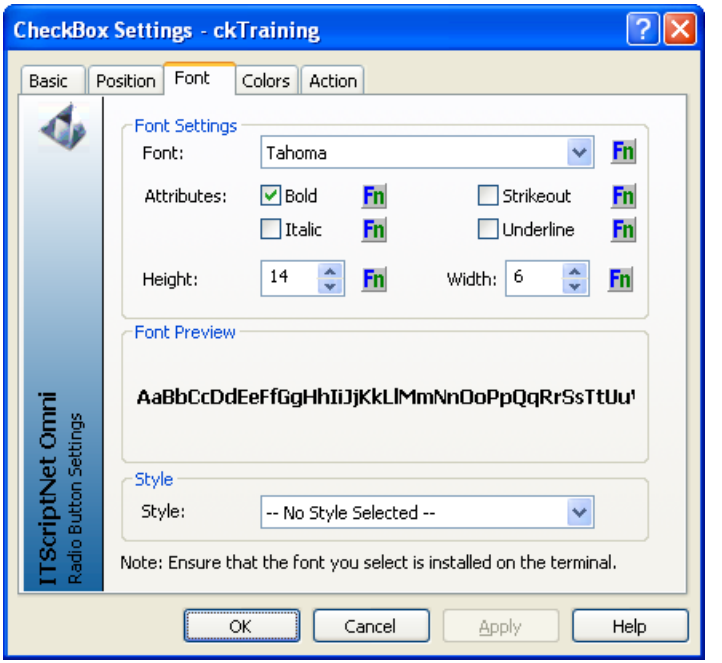


Top Position

The Top Position Setting specifies the vertical location in pixels of the upper-left corner of the Checkbox element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Top Position Setting is set to 0, the Checkbox will be all the way to the top of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the top position to a value larger than the screen the element will be located off the prompt screen

Font

The Font Tab on the Checkbox element Settings screen allows you to customize standard Checkbox elements.



Font

The Font selection allows you to choose a font for the Checkbox text. Note that if you select a font that is not installed on your portable terminal, the operating system of the terminal will convert the font to a font that is installed. The effect of this conversion will most likely be that the positioning of the font for the Checkbox on the terminal will not be consistent with the design view. The area under the font selection displays a preview of the font you have chosen.

Attributes

The Attributes section allows you to include Bold, Italic, Underline, or Strikeout attributes to your font choices. A check next to the attribute turns that attribute on.

Height & Width

The Height and Width Font Settings specify the Height and Width of the Font size in pixels. The Font Preview section will reflect the Height and Width specified. If you use a Font Width of zero, Windows will select an appropriate default width.

Style

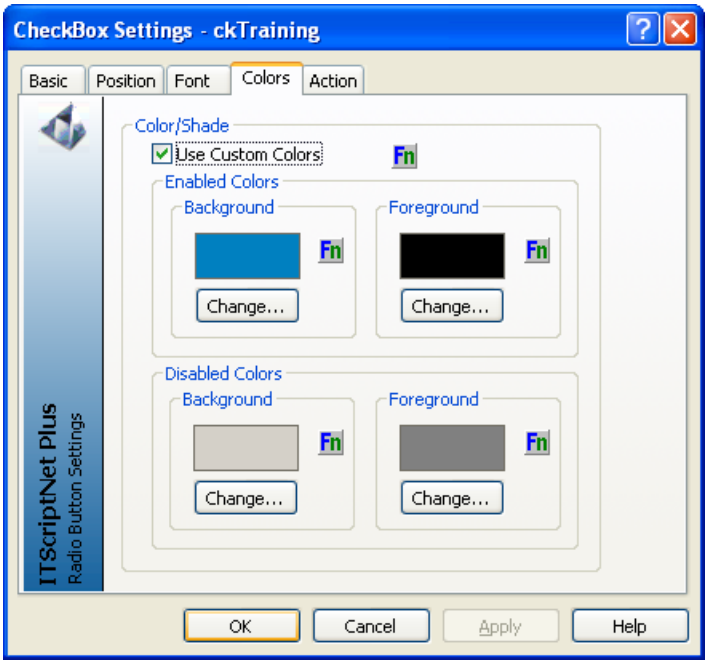
The Style setting allows you to apply a pre-defined Font and Color set to this element. The Style must have been created with the Style Editor on the Program menu. Any changes made to the Style in the Style Editor will automatically be applied to this element. If you change any Font or Color setting on the element, the Style will be removed.

Colors

The Colors tab adds another dimension of flexibility and customization to the design of Checkbox elements. The colors will only be effective for portable terminals that have a color display. The first step towards customizing the colors of a standard button is to check the 'Use Custom Colors' option on the Colors tab. Once checked, the colors specified on the tab will be used.

Enabled Colors

The Background color and the Foreground (Text) color on the Checkbox can be specified independently. The Enabled Colors are used when the element is enabled. Click on the Change… button to bring up the Color screen. You can select from the basic colors on the left, select a color from the color matrix on the right, specify Hue/Saturation/Luminosity, or specify Red, Blue, and Green values to specify a custom color.



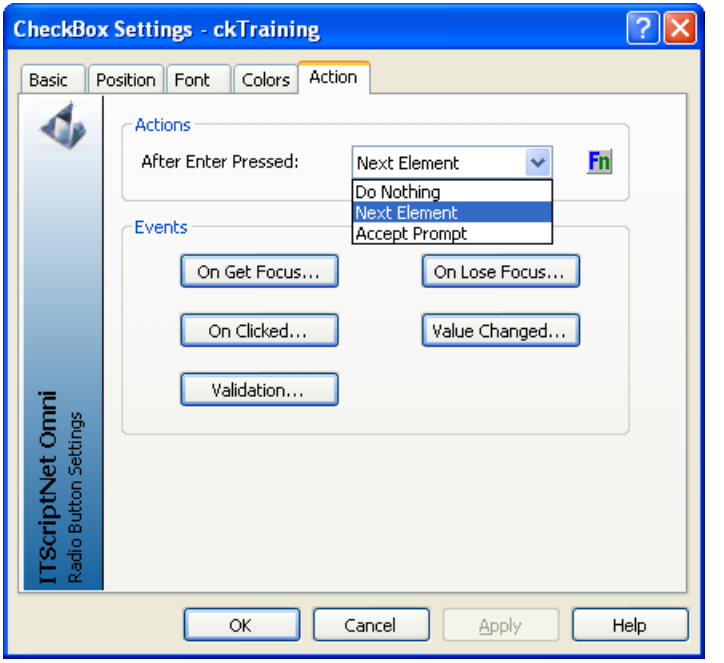
Disabled Colors

The Background color and the Foreground (Text) color on the buttons can be specified independently. The disabled colors are used when the button is disabled and not able to be clicked. Note that some Windows CE and PocketPC devices may override the Disable text color.

Action

The Action tab contains the settings to control the behavior of the Checkbox element after a response has been registered and if the user presses the Enter button. This tab also provides access to the in-prompt scripts that can be used to customize behavior when the element gets or loses the focus or when the Checkbox element is clicked.

After Enter Pressed



When a user enters a response for the Checkbox element, it is often desirable to have the data collection program automatically advance to the next element so that the user does not have to click on the next element to keep entering data. To help minimize the number of clicks required for the user, it is often a good idea to take advantage of the After Enter Pressed setting. There are 3 choices for how the element will behave after the Enter button is pressed. The default behavior is for the element to do nothing. In this case, pressing Enter will toggle the checked state of the checkbox. You can also choose to have the element advance the focus to the next element on the prompt. The order of the elements is determined on the prompt's settings screen. The last choice is to have the Enter key act as the acceptance for the entire prompt and thus the program advances to the next prompt as defined on the prompt's settings screen.

Events

The Check Box element has several special event-driven scripts that allow for customized behavior.

- The **OnGetFocus** script runs when the element receives the focus. This can happen if the user clicks on the element, tabs to the element, if the focus is set to the element in a script, or can occur when a prompt is displayed if the element is the first element on the prompt.
- The **OnLoseFocus** Script runs when the element loses the focus, or in other words, when the focus is moved to another element.
- The **OnClick** Script runs whenever the element is tapped or clicked.
- The **ValueChanged** Script runs whenever the data in the control is changed, whether by data entry or assigning from a script.
- The **Validation** Script runs when the prompt is accepted, as part of the prompt validation process. If you call the *ValidationFail* function from within this script, the prompt validation will fail, the error message specified will be displayed, and the focus will be set back to this element.

Combobox Element

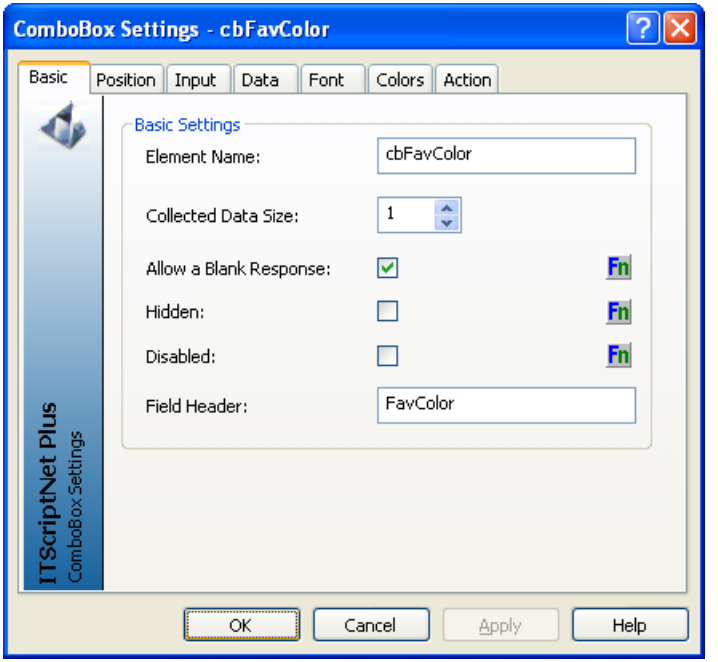
The Combobox element is the drop-down selection list for data collection that is useful in situations where you want your user to select one item from a list of choices. That list of choices can be a pre-defined list from design time (see Data Tab section) or can come from a validation file.

To add a Combobox element to a prompt, click on the Combobox 

from the element list on the right side of the main Generator Application screen. If you cannot see the element list, you can turn it on by checking the Elements menu under the View main menu item. After you click the Combobox element, the Combobox Settings screen will be displayed. You will need to specify the settings for your Combobox. The key settings are the Element Name, the collected data size, and whether the data for the Combobox will be pre-defined (static) or come from a validation file. These and all Combobox Settings are discussed below.

Basic Tab

The Basic Tab on the Combobox Settings screen is shown here.



Element Name

Combobox elements, like all elements must have a name. When you first create the Combobox element the name will be "ComboBox1". You may change the name to another valid element name. Element names can be up to 20 characters in length and must be unique within the prompt. The following characters may not be used on Input Text element names: # $ @ _ - * . / = < > ( ) & as they are reserved.

Collected Data Size

The collected data size setting is the size of the underlying values that are associated with each choice in the list of items in the Combobox. Each Combobox item has a defined display text and a defined value that is stored if that item is selected. The text to display and underlying data values are defined on the Data tab.

Allow a Blank Response

This setting will determine whether or not to allow a blank response for the Combobox element. By default this setting is turned on, meaning that a response is not required. If the box is checked, thus allowing a blank response, the user collecting the data will be able to go to the next prompt

even if he has not chosen one of the Combobox items and therefore the Combobox element has no response and is blank. Allowing a blank response is useful for optional data in a data collection program.

Hidden, Disabled

The Hidden setting will cause the Combobox element and all its items to be hidden when checked. The Disabled setting will cause the Combobox element to be disabled. A response cannot be entered, meaning the Combobox options cannot be clicked or scrolled, if the Combobox element is hidden and/or disabled.

Database Field/Column Header/Field Header

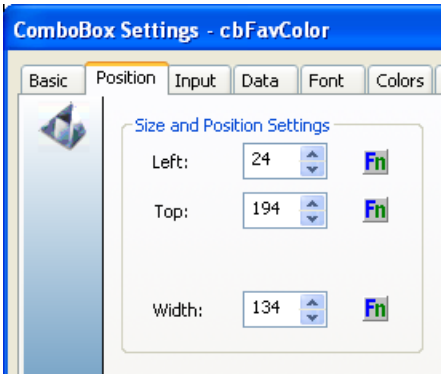
This setting specifies where the data collected for this element is stored after it is sent to the PC and processed. Depending on the settings specified in the *Configure Receive* screen, this field may have different meanings.

If the program is configured to send its data to a Microsoft Access file or an ODBC data source, the target database field must be set up properly so the data collected can be sent to the desired field. A field from the target database must be selected from the list of fields. The fields in the list are the fields defined for the database and table that are established on the *Configure Receive* screen. The <Ignore> selection can be chosen and the collected data for the Combobox element will be ignored when the collected data is sent to the database. Fields are output to Access and ODBC if the field specified here exists and the maximum length of the response is not zero.

If the program is configured to send its data to a Microsoft Excel file, this setting will become the column header for the data. For Excel files, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen.

If the program is configured to send its data to a text file with a header, this setting will become the field header for the data in the output file. For text files with headers, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen. If the program is configured to send its data to a text file with no header information, then there will be no Field Header setting on the Combobox Settings screen.

Position Tab



The Position Tab on the Combobox Settings screen controls the size and position of the Combobox element on the prompt. You can also control the size and position of the Combobox element from the main design window. You can move the Combobox by selecting it and holding the mouse while you move the Combobox to the desired location. You can also resize the Combobox by selecting it and hovering over the resize boundaries of the Combobox element and dragging the Combobox to resize it.

Left Position

The Left Position Setting specifies the horizontal location in pixels of the upper-left corner of the Combobox. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Left Position Setting is set to 0, the Combobox will be all the way to the left of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the left position to a value larger than the screen the element will be located off the prompt screen.

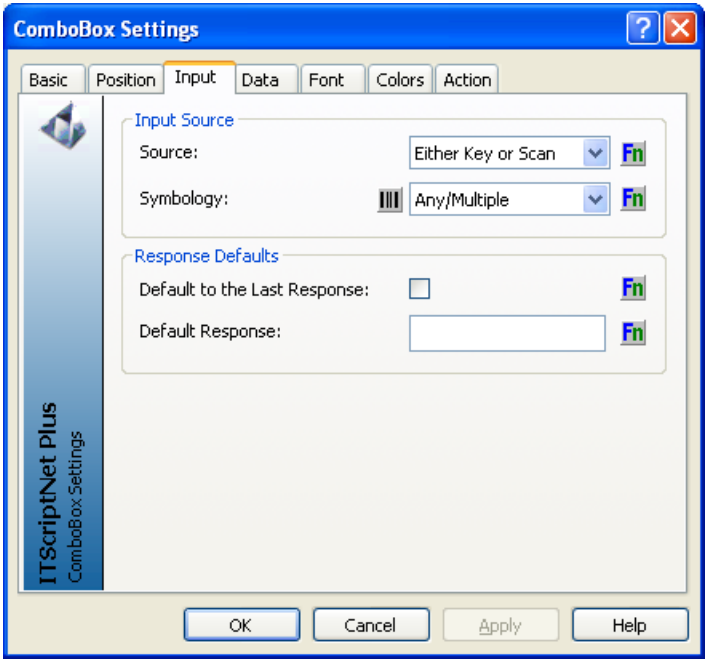
Top Position

The Top Position Setting specifies the vertical location in pixels of the upper-left corner of the Combobox. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Top Position Setting is set to 0, the Combobox will be all the way to the top of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the top position to a value larger than the screen the element will be located off the prompt screen.

Width

The Width Setting specifies the width in pixels of the Combobox.

Input Tab

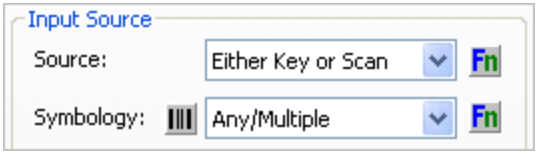


Input Source

The setting for Source specifies whether the response to the Combobox can be entered only on the terminal's keypad, or by either keypad or scan. By default, Combo-boxes are set to allow either keypad or scan for data entry. Most likely users will select an item from the Combobox using the arrow keys or by tapping on the screen. Depending on the application, requiring barcode scans will *reduce data collection time* (scanning is faster than punching in the data on a keypad) and *maximize accuracy*.

Symbology

The setting for Symbology is used to limit the acceptable barcode symbologies when scanning a response for a Combobox element. There are many different barcode symbologies. A barcode symbology is an algorithm for encoding data into a barcode. Some of the more common symbologies are: Code 39, I2of5, Code 128, and UPCA. The Combobox element can be set to accept any barcode symbology (the terminal has to be able to decode the symbology, please refer to documentation on the specific portable terminal you are using for a list of symbologies compatible with your hardware). The Any/Multiple selection will allow a user to scan any symbology that is configured as a default for the scanner/imager for your hardware. The Any/Multiple setting is the default for any new Combobox element that you create.



The Symbology property can also be set to accept only one type of barcode symbology. To select a single symbology, select it from the list. The default settings for the symbology on your terminal will be used. By way of example, let's say that your warehouse uses Code 39 barcodes for your part numbers. However, the boxes also contain UPC codes for those products. When collecting data, you want to insure that the Code 39 barcode is scanned and not the UPC code. By setting the Combobox element to only accept Code 39, the program will reject the UPC codes and any other symbology. Specifying a specific symbology is a means of validating your data and *maximizing data accuracy*.

Depending on your data collection needs, you may need to customize the Symbology setting. In our example above, the warehouse had Code 39 and UPC part numbers. Let's say that now you need to be able to scan either a Code 39 or a UPC, but still do not want to allow any other symbologies. You can do this by clicking on the button with the barcode icon to bring up the *Advanced Symbology Settings* screen. Please refer to the section in this User Guide describing the *Advanced Symbology Settings* screen. The *Advanced Symbology Settings* screen will also allow you to control each parameter available for one or more symbologies. If you do not customize the barcode symbology settings, the default settings for the single symbology specified or all symbologies will be used. The defaults are appropriate for most data collection applications, but if you need to control the settings more precisely, **IT*Script*Net** allows you access and full control of the symbologies. If the *Advanced Symbology Settings* screen is used to customize the symbology behavior for a Combobox, the barcode icon will change colors to provide an easy visual indicator that customized symbology settings are in use.

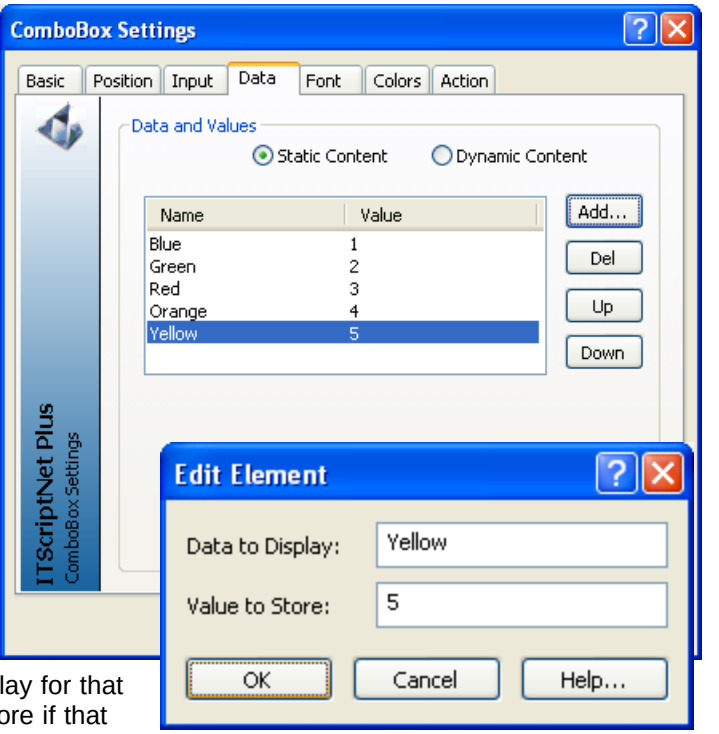
Response Defaults

The Response Defaults settings can set the default response for the element to a specific value or to the value of the response for this element from the last data record collected. When creating a new Combobox element, the response default is initially set to no default. When in this state, the user will have to enter data or select an item from the Combobox each time through the prompting loop during data collection.

Using response defaults, the *Default to the Last Response* option can be enabled and will cause the response to default to the previous response for the element. The user will still be able to change the response, but can save time by accepting the default. This feature is useful when an element will often have the same information for several prompting loops.

The other option available for the response default is to actually set a response default that will always be shown as the default response for the Combobox. It is important to note that the value of the default response specified must match one of the values of an item in the Combobox – not the text that is shown in the Combobox.

Data Tab



The text that is displayed to the user for each Combobox item (choice) is defined here as well as the underlying value of the Combobox item that is stored as collected data corresponding for the Combobox element. There are two types of data and values: Static and Dynamic. Static content requires that the data is entered at design time into the list on the Data tab. For dynamic content, the Combobox display text and values come from a validation file.

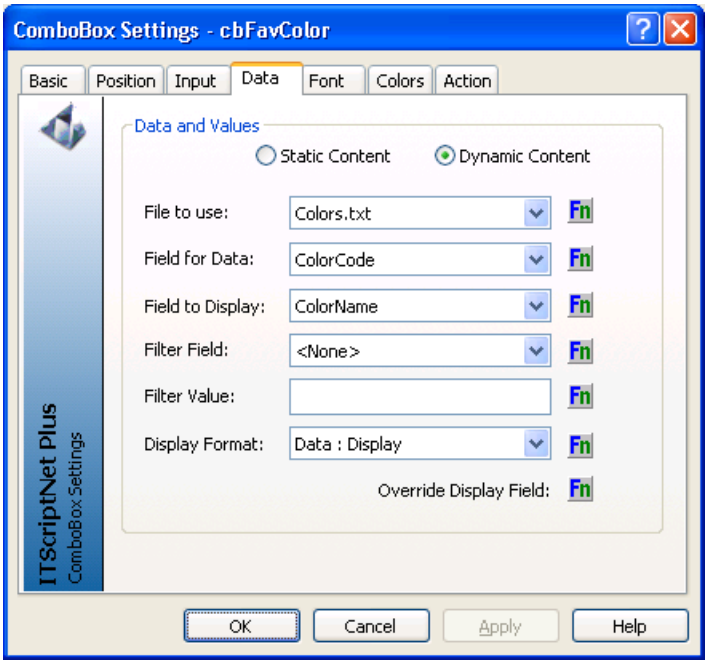
Static Content

For Static content, each item in the list of items for the Combobox must be entered into the Data Tab. Click the Add… button to bring up the Edit Element box. For each Combobox choice, you must define the data to display for that Combobox item and also the value to store if that choice is selected. As you add Combobox items, they will be added to the list that is displayed on the Data tab. Double-clicking an item in the list will bring up the Edit Element screen for editing. The Value to Store for each item should correspond to the Collected Data Size setting on the Basic tab. In the example shown, there is a list of colors defined as choices for this Combobox element. Each color that will be in the list for the user to see has an underlying data value that actually is stored. Blue has a value of 1, Green has 2, etc.

The Delete button removes the selected Combobox item from the list. The Up and Down buttons allow you to manipulate the order of the list of Combobox choices by moving the item up or down in the list.

Dynamic Content

Validation files can be used to fill a Combobox element dynamically at run-time instead of at design time. This gives your data collection program more flexibility.



The "File to use" drop-down box allows you to select a validation file to use to fill the Combobox. The validation text file must be defined in the *Validation Files* screen before a Combobox can be configured to use the validation text file. Please refer to the *Validation Files* section of this User Guide for more information on setting up the validation file for your data collection program.

Once the validation file has been selected, the Field for Data drop-down list will be filled with the fields in the validation file. Select a field to use as the underlying data in the Combobox that will be stored for the item if the item is chosen by the user when collecting data. The field selected for the Field to Display setting will determine what field from the validation file is used as the displayed text in the Combobox.

The *Filter Field* is used to apply a filter to the validation file so that only matching records are added to the picklist. For example, if you were filling a Picklist from an Order Items file, you might specify an Order Number field as the filter field, and a specific order number as the Filter Value so that only items on that order are listed in the Picklist. The *Filter Value* is the value that should be used to filter the validation file. You can enter a value, or use the In-Prompt Script to override the Filter Value so you can filter by the value of another prompt or variable.

The *Display Format* allows you to control how data is displayed in the Picklist. The standard format is to display both the Data (the Validation Field) and the Display (the Lookup Field), in the format Data : Display. You can also specify to display just the Data field, or just the Display field.

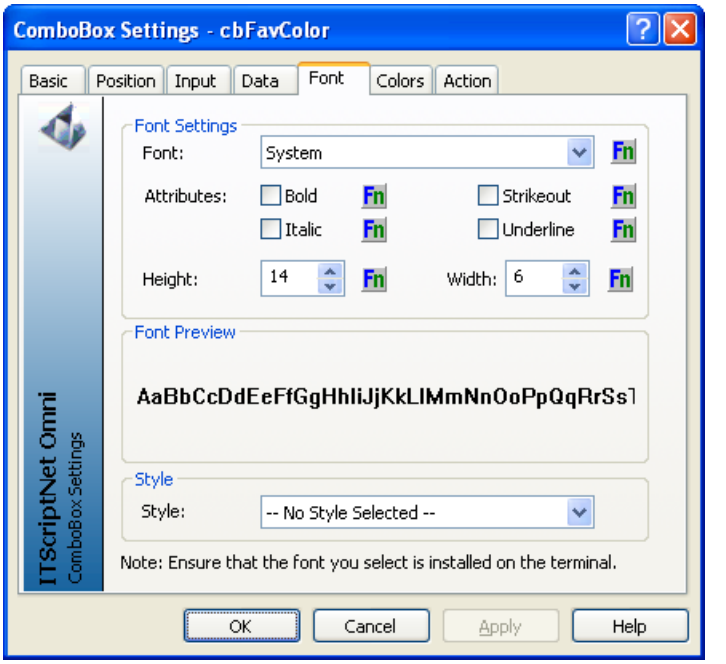
The *Override Display Field* allows you to control the way data is presented in the picklist even further than the Filter and Display options already described. This script is executed once for each record in the validation file. The result of the script is used as the text to display in the picklist, overriding the default display. If the script returns an empty string, the record will not be added to

the picklist. You can reference the fields in the validation file record using the *PickListField* function. This allows you to completely control the way data is presented in a Picklist.

The example shown uses a validation file called *colors.txt*. There is a field called ColorID, which contains a numeric code for each color listed. The names of the colors are in the field named ColorDesc and will be displayed to the user in the Combobox. This example provides exactly the same results as shown in the Static Content section.

Font Tab

The Font Tab on the Combobox Settings screen allows you to customize standard Combobox elements.



Font

The Font selection allows you to choose a font for the Combobox text. Note that if you select a font that is not installed on your portable terminal, the operating system of the terminal will convert the font to a font that is installed. The effect of this conversion will most likely be that the positioning of the font for the Combobox on the terminal will not be consistent with the design view. The area under the font selection displays a preview of the font you have chosen.

Attributes

The Attributes section allows you to include Bold, Italic, Underline, or Strikeout attributes to your font choices. A check next to the attribute turns that attribute on.

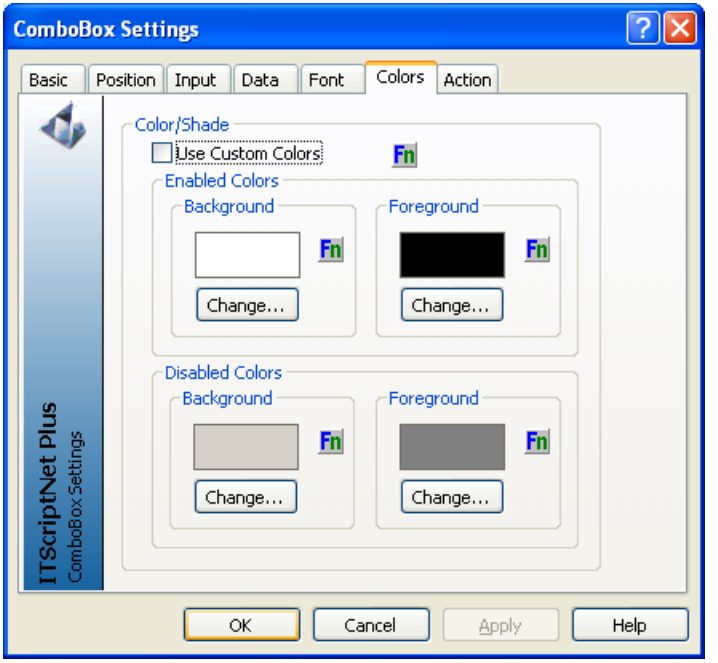
Height & Width

The Height and Width font settings specify the Height and Width of the font size in pixels. The Font Preview section will reflect the Height and Width specified. If you use a Font Width of zero, Windows will select an appropriate default width.

Style

The Style setting allows you to apply a pre-defined Font and Color set to this element. The Style must have been created with the Style Editor on the Program menu. Any changes made to the Style in the Style Editor will automatically be applied to this element. If you change any Font or Color setting on the element, the Style will be removed.

Colors Tab



The Colors tab adds another dimension of flexibility and customization to the design of Combobox elements. The colors will only be effective for portable terminals that have a color display. The first step towards customizing the colors is to check the 'Use Custom Colors' option on the Colors tab. Once checked, the colors specified on the tab will be used.

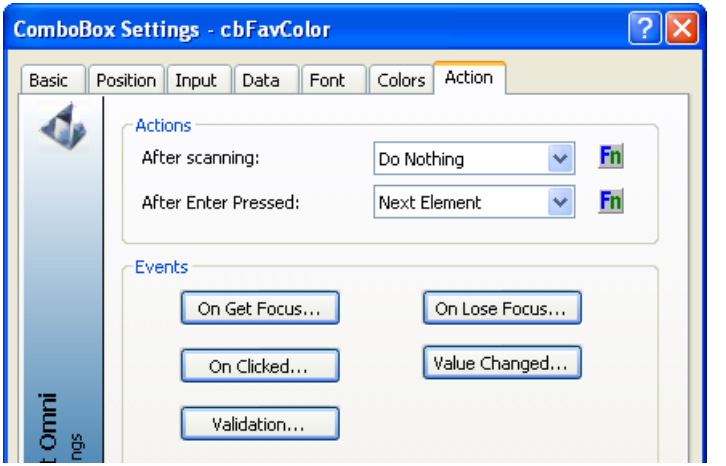
Enabled Colors

The Background color and the Foreground (Text) color on the Combobox can be specified independently. The enabled colors are used when the element is enabled. Click on the Change… button to bring up the Color screen. You can select from the basic colors on the left, select a color from the color matrix on the right, specify Hue/Saturation/Luminosity, or specify Red, Blue, and Green values to specify a custom color.

Disabled Colors

The Background color and the Foreground (Text) color on the buttons can be specified independently. The disabled colors are used when the button is disabled and not able to be clicked. Note that Windows CE or PocketPC may override the disabled foreground color.

Action Tab

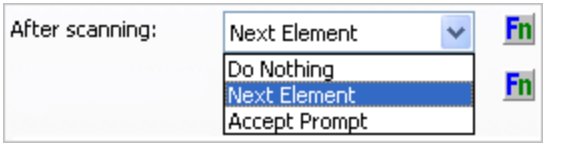


The Action tab contains the settings to control the behavior of the Combobox element after a response has been scanned and/or if the user presses the Enter button. This tab also has access to the in-prompt scripts that can be used to customize behavior when the element gets or loses the focus.

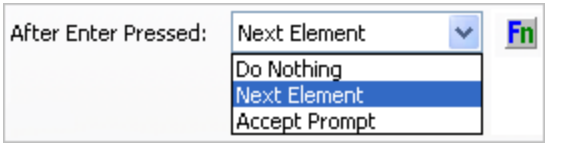
After Scanning

When a user scans a response for the Combobox element (as long as scanning is configured as a valid input source on the Input tab), the After Scanning setting determines how the element will behave. There are 3 choices. The default behavior is for the element to do nothing after the scan. You can also choose to have the element advance the focus to the next element on the prompt. The order of the elements is determined on the prompt's settings screen. The last choice is to have the scan act as the acceptance for the entire prompt and thus the program advances to the next prompt as defined on the prompt's settings screen.

After Enter Pressed



When a user enters a response for the Combobox element, it is often desirable to have the data collection program automatically advance to the next element so that the user does not have to click on the next element to keep entering data. To help minimize the number of clicks required for the user, it is often a good idea to take advantage of the After Enter Pressed setting. There are 3 choices for how the element will behave after the Enter button is pressed. The default behavior is for the element to do nothing. You can also choose to have the element advance the focus to the next element on the prompt. The order of the elements is determined on the prompt's settings screen. The last choice is to have the Enter key act as the acceptance for the entire prompt and thus the program advances to the next prompt as defined on the prompt's settings screen.



Events

The Input Text element has several special event-driven scripts that allow for customized behavior.

- The **OnGetFocus** script runs when the element receives the focus. This can happen if the user clicks on the element, tabs to the element, if the focus is set to the element in a script, or can occur when a prompt is displayed if the element is the first element on the prompt.
- The **OnLoseFocus** Script runs when the element loses the focus, or in other words, when the focus is moved to another element.
- The **OnClick** Script runs whenever the element is tapped or clicked.
- The **ValueChanged** Script runs whenever the data in the control is changed, whether by scanning, data entry, or assigning from a script.
- The **Validation** Script runs when the prompt is accepted, as part of the prompt validation process. If you call the *ValidationFail* function from within this script, the prompt validation will fail, the error message specified will be displayed, and the focus will be set back to this element.

Listbox Element

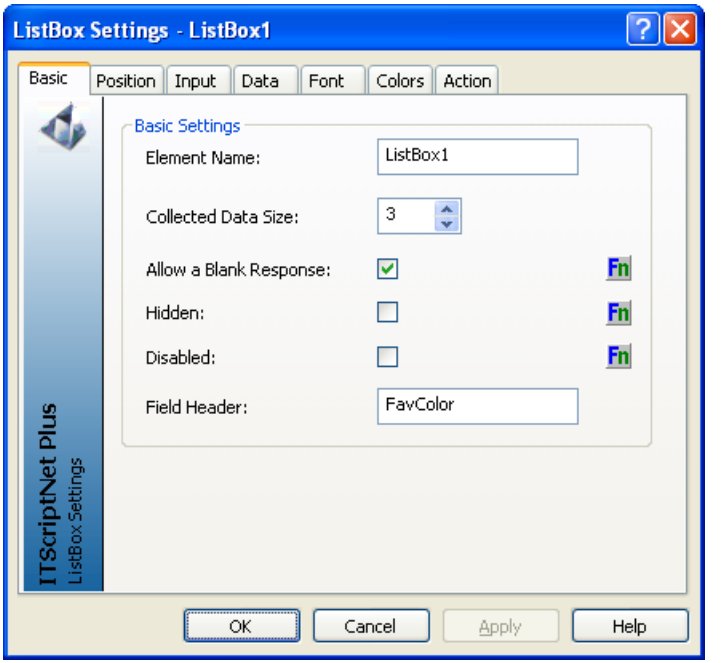
The Listbox element is the selection list for data collection that is useful in situations where you want your user to select one item from a list of choices. That list of choices can be a pre-defined list from design time or can come from a validation file.

To add a Listbox element to a prompt, click on the Listbox 

from the element list on the right side of the main Generator Application screen. If you cannot see the element list, you can turn it on by checking the Elements menu under the View main menu item. After you click the Listbox element, the Listbox Settings screen will be displayed. You will need to specify the settings for your Listbox. The key settings are the element Name, the collected data size, and whether the data for the Listbox will be pre-defined (static) or come from a validation file. These and all Listbox Settings are discussed below.

Basic Tab

The Basic tab on the Listbox Settings screen is shown here.



Element Name

Listbox elements, like all elements, must have a name. When you first create the Listbox element the name will be displayed as "Listbox1". You can change the element name to a valid element name. The name of the Listbox element can be up to 20 characters in length and must be a unique name within the prompt. The following characters may not be used on Input Text element names: # $ @ _ - * . / = < > ( ) & as they are reserved.

Collected Data Size

The Collected Data Size setting is the size of the underlying values that are associated with each choice in the list of items in the Listbox. Each Listbox item has a defined display text and a defined value that is stored if that item is selected. The text to display and underlying data values are defined on the Data tab. Please also refer to the section in this User Guide that describes the Data tab of the Listbox Settings screen.

Allow a Blank Response

This setting will determine whether or not to allow a blank response for the Listbox element. By default this setting is turned on, meaning that a response is not required. If the box is checked,

thus allowing a blank response, the user collecting the data will be able to go to the next prompt even if he has not chosen one of the Listbox items and therefore the Listbox element has no response and is blank. Allowing a blank response is useful for optional data in a data collection program.

Hidden, Disabled

The Hidden setting will cause the Listbox element and all its items to be hidden when checked. The Disabled setting will cause the Listbox element to be disabled. A response cannot be entered, meaning the Listbox options cannot be clicked or scrolled, if the Listbox element is hidden and/or disabled.

Database Field/Column Header/Field Header

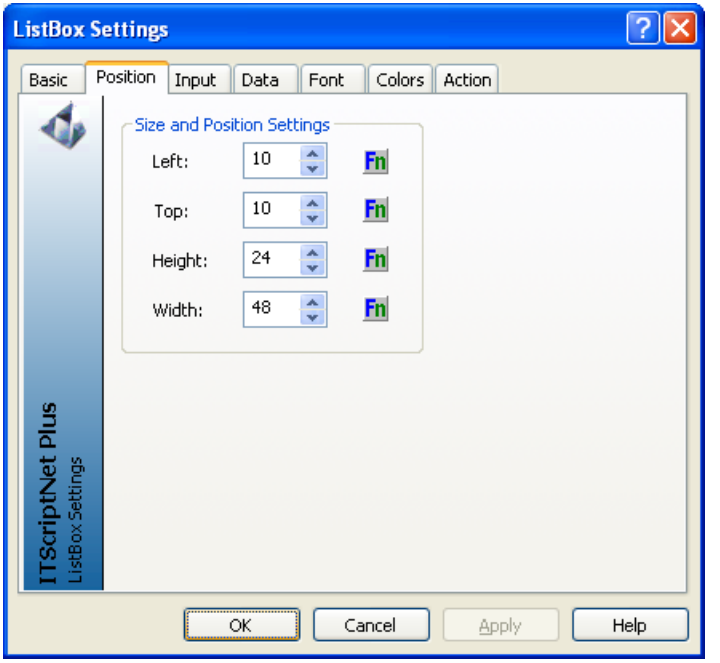
This setting specifies where the data collected for this element is stored after it is sent to the PC and processed. Depending on the settings specified in the *Configure Receive* screen, this field may have different meanings.

If the program is configured to send its data to a Microsoft Access file or an ODBC data source, the target database field must be set up properly so the data collected can be sent to the desired field. A field from the target database must be selected from the list of fields. The fields in the list are the fields defined for the database and table that are established on the *Configure Receive* screen. The <Ignore> selection can be chosen and the collected data for the element will be ignored when the collected data is sent to the database. Fields are output to Access and ODBC if the field specified here exists and the maximum length of the response is not zero.

If the program is configured to send its data to a Microsoft Excel file, this setting will become the column header for the data. For Excel files, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen.

If the program is configured to send its data to a text file with a header, this setting will become the field header for the data in the output file. For text files with headers, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen. If the program is configured to send its data to a text file with no header information, then there will be no Field Header setting on the Listbox Settings screen.

Position Tab



The Position Tab on the Listbox Settings screen controls the size and position of the Listbox element on the prompt. You can also control the size and position of the Listbox element from the main design window. You can move the Listbox by selecting it and holding the mouse while you move the Listbox to the desired location. You can also resize the Listbox by selecting it and hovering over the resize boundaries of the Listbox element and dragging the Listbox to resize it.

Left Position

The Left Position setting specifies the horizontal location in pixels of the upper-left corner of the Listbox. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Left Position setting is set to 0, the Listbox will be all the way to the left of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the left position to a value larger than the screen the element will be located off the prompt screen.

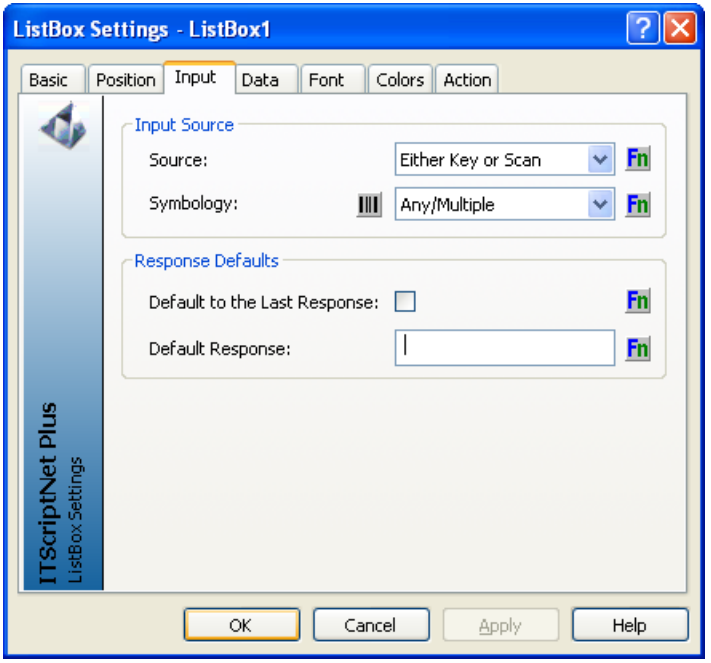
Top Position

The Top Position setting specifies the vertical location in pixels of the upper-left corner of the Listbox. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Top Position setting is set to 0, the Listbox will be all the way to the top of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the top position to a value larger than the screen the element will be located off the prompt screen.

Height & Width

The Height setting specifies the height, in pixels, of the Listbox element. The Width setting specifies the width in pixels of the Listbox.

Input Tab

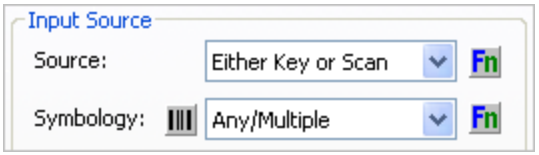


Input Source

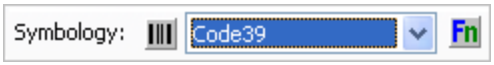
The setting for Source specifies whether the response to the Listbox can be entered only on the terminal's keypad or by either scan or keypad. By default, Listboxes are set to allow either keypad or scan for data entry. Typically, users will select an item from the Listbox using the arrow keys or by tapping on the screen. Depending on the application, allowing barcode scans can *reduce data collection time* (scanning is faster than punching in the data on a keypad) and *maximize accuracy*.

Symbology

The setting for Symbology is used to limit the acceptable barcode symbologies when scanning a response for a Listbox element. There are many different barcode symbologies. A barcode symbology is an algorithm for encoding data into a barcode. Some of the more common symbologies are: Code 39, I2of5, Code 128, and UPCA. The Listbox element can be set to accept any barcode symbology (the terminal has to be able to decode the symbology, please refer to documentation on the specific portable terminal you are using for a list of symbologies compatible with your hardware). The Any/Multiple selection will allow a user to scan any symbology that is configured as a default for the scanner/imager for your hardware. The Any/Multiple setting is the default for any new Listbox elements that you create.



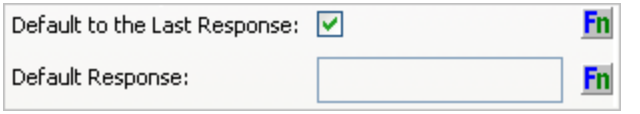
The Symbology property can be set to accept only one type of barcode symbology. To select a single symbology, select it from the list. The default settings for the symbology on your terminal will be used. By way of example, let's say that your warehouse uses Code 39 barcodes for your part numbers. However, the boxes also contain UPC codes for those products. When collecting data, you want to insure that the Code 39 barcode is scanned and not the UPC code. By setting the Listbox element to only accept Code 39, the program will reject the UPC codes and any other symbology. Specifying a specific symbology is a means of validating your data and *maximizing data accuracy*.



Depending on your data collection needs, you may need to customize the Symbology setting. In our example above, the warehouse had Code 39 and UPC part numbers. Let's say that now you need to be able to scan either a Code 39 or a UPC, but still do not want to allow any other symbologies. You can do this by clicking on the button with the barcode icon to bring up the *Advanced Symbology Settings* screen. Please refer to the section in this User Guide describing the *Advanced Symbology Settings* screen. The *Advanced Symbology Settings* screen will also allow you to control each parameter available for one or more symbologies. If you do not customize the barcode symbology settings, the default settings for the single symbology specified or all symbologies will be used. The defaults are appropriate for most data collection applications, but if you need to control the settings more precisely, **IT*Script*Net** allows you access and full control of the symbologies. If the *Advanced Symbology Settings* screen is used to customize the symbology behavior for a Listbox, the barcode icon will change colors to provide an easy visual indicator that customized symbology settings are in use.

Response Defaults

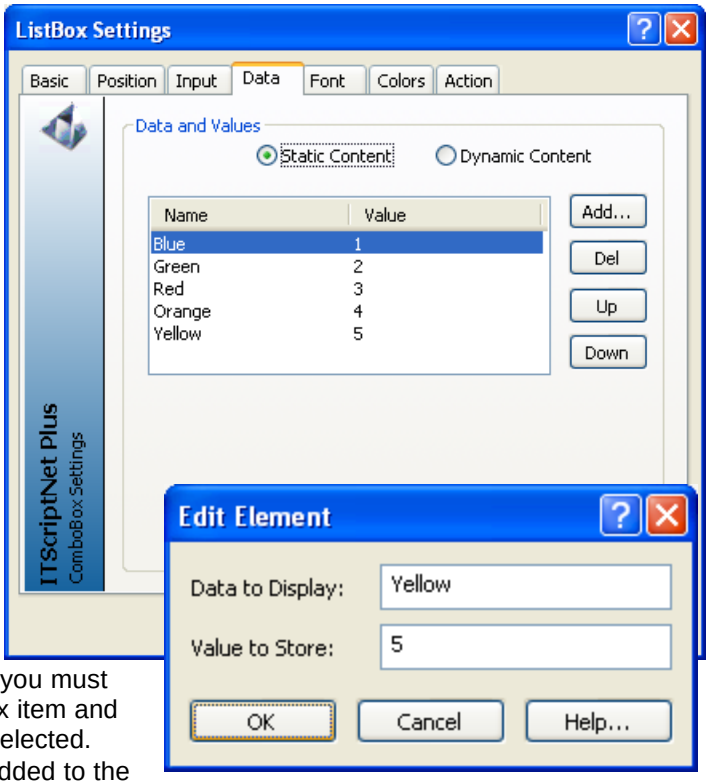
The Response Defaults settings can set the default response for the element to a specific value or the value of the response from the last data record collected for the element. When creating a new Listbox element, the response default is initially set to no default. When in this state, the user will have to enter data or select an item from the Listbox each time through the prompting loop during data collection.



Using response defaults, the *Default to the Last Response* option can be enabled and will cause the response to default to the previous response. The user will still be able to change the response, but can save time by accepting the default. This feature is useful when an element will often have the same information for several prompting loops.

The other option available for the response default is to actually set a response default that will always be shown as the default response for the Listbox. It is important to note that the value of the default response specified must match one of the values of an item in the Listbox – not the text that is shown in the Listbox.

Data Tab



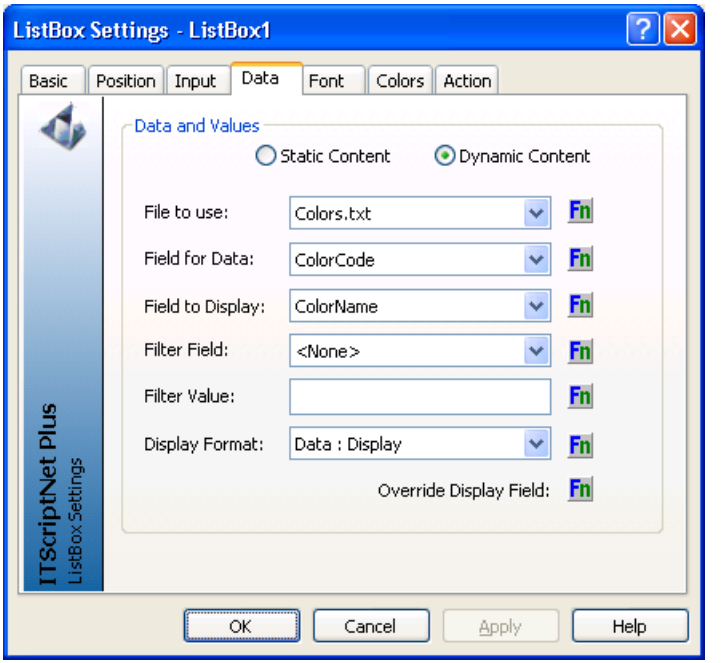
This tab is where you specify the Listbox items. The text that is displayed to the user for each Listbox item (choice) is defined here as well as the underlying value of the Listbox item that is stored as collected data corresponding to the Listbox element. There are two types of data and values: Static and Dynamic. Static content requires that the data be entered at design time into the list on the Data tab. For dynamic content, the Listbox display text and values come from a validation file.

Static Content

For Static Content, each item in the list of items for the Listbox must be entered into the Data tab. Click the Add… button to bring up the Edit Element box. For each Listbox choice, you must define the data to display for that Listbox item and also the value to store if that choice is selected. As you add Listbox items, they will be added to the list that is displayed on the Data tab. Double-clicking an item in the list will bring up the Edit Element screen for editing. The Value to Store for each item should correspond to the Collected Data Size setting on the Basic tab. In the example shown, there is a list of colors defined as choices for this Listbox element. Each color that will be in the list for the user to see has an underlying data value that actually is stored. Blue has a value of 1, Green has 2, etc.

The Del button removes the selected Listbox item from the list. The Up and Down buttons allow you to manipulate the order of the list of Listbox choices listed. The Up and Down buttons move the selected Listbox item either Up or Down.

Dynamic Content



Validation files can be used to fill a Listbox element dynamically at run-time instead of at design time. This gives your data collection program more flexibility.

The 'File to use' drop-down box allows you to select a validation file to use to fill the Listbox. The validation text file must be defined from the *Validation Files* screen before a Listbox can be configured to use the validation text file. Please refer to the *Validation Files* Screen section of this User Guide for more information on setting up the validation file for your data collection program.

Once the validation file has been selected, the Field for Data drop-down list will be filled with the fields in the validation file. Select a field to use as the underlying data in the Listbox that will be stored for the item if the item is chosen by the user when collecting data. The Field selected for the Field to Display setting will determine what field from the validation file is used as the displayed text in the Listbox.

The *Filter Field* is used to apply a filter to the validation file so that only matching records are added to the picklist. For example, if you were filling a Picklist from an Order Items file, you might specify an Order Number field as the filter field, and a specific order number as the Filter Value so that only items on that order are listed in the Picklist. The *Filter Value* is the value that should be used to filter the validation file. You can enter a value, or use the In-Prompt Script to override the Filter Value so you can filter by the value of another prompt or variable.

The *Display Format* allows you to control how data is displayed in the Picklist. The standard format is to display both the Data (the Validation Field) and the Display (the Lookup Field), in the format Data : Display. You can also specify to display just the Data field, or just the Display field.

The *Override Display Field* allows you to control the way data is presented in the picklist even further than the Filter and Display options already described. This script is executed once for each record in the validation file. The result of the script is used as the text to display in the picklist, overriding the default display. If the script returns an empty string, the record will not be added to

the picklist. You can reference the fields in the validation file record using the *PickListField* function. This allows you to completely control the way data is presented in a Picklist.

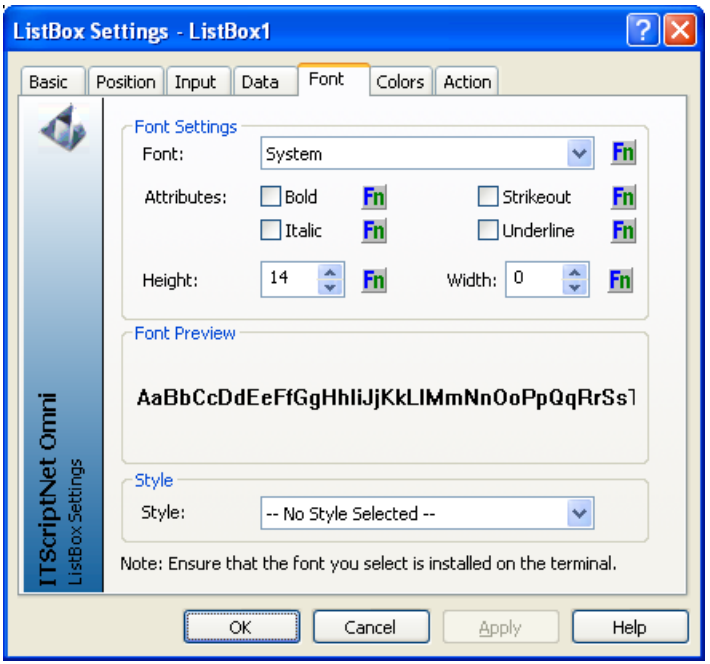
The example shown uses a validation file called *colors.txt*. There is a field called ColorCode, which contains a numeric code for each color listed. The names of the colors are in the field named ColorName and will be displayed to the user in the Listbox. This example provides exactly the same results as shown in the Static Content section.

Font Tab

The Font tab on the Listbox Settings screen allows you to customize standard Listbox elements.

Font

The Font selection allows you to choose a font for the Listbox. Note that if you select a font that is not installed on your portable terminal, the operating system of the terminal will convert the font to a font that is installed. The effect of this conversion will most likely be that the positioning of the font for the Listbox on the terminal will not be consistent with the design view. The area under the font selection displays a preview of the font you have chosen.



Attributes

The Attributes section allows you to include Bold, Italic, Underline, or Strikeout attributes to you font choices. A check next to the attribute turns that attribute on.

Height & Width

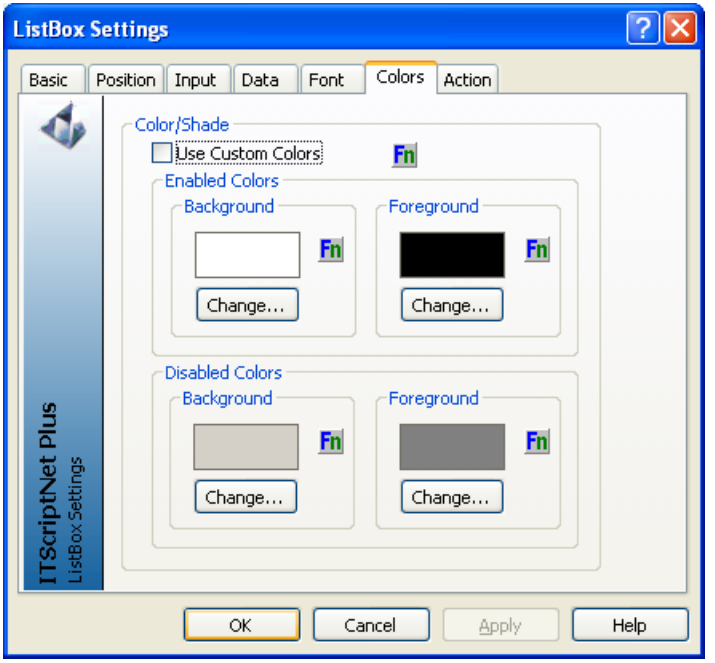
The Height and Width font settings specify the Height and Width of the font size in pixels. The Font Preview section will reflect the Height and Width specified. If you use a Font Width of zero, Windows will select an appropriate default width.

Style

The Style setting allows you to apply a pre-defined Font and Color set to this element. The Style must have been created with the Style Editor on the Program menu. Any changes made to the Style in the Style Editor will automatically be applied to this element. If you change any Font or Color setting on the element, the Style will be removed.

Colors Tab

The Colors tab adds another dimension of flexibility and customization to the design of Listbox elements. The colors will only be effective for portable terminals that have a color display. The first step towards customizing the colors of a standard button is to check the 'Use Custom Colors' option on the Colors tab. Once checked, the colors specified on the tab will be used.



Enabled Colors

The Background color and the Foreground (Text) color on the element can be specified independently. The enabled colors are used when the element is enabled. Click on the Change… button to bring up the Color screen. You can select from the basic colors on the left, select a color from the color matrix on the right, specify Hue/Saturation/Luminosity, or specify Red, Blue, and Green values to specify a custom color.

Disabled Colors

The Background color and the Foreground (Text) color on the buttons can be specified independently. The disabled colors are used when the button is disabled and not able to be clicked. Depending on the terminal, the disabled colors may be overridden by the operating system of the terminal and the disabled colors may not be displayed as designed.

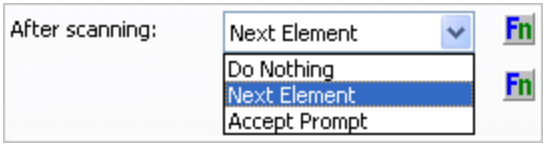
Action

The Action tab contains the settings to control the behavior of the Listbox element after a response has been scanned and/or if the user presses the Enter button. This tab also has access to the in-prompt scripts that can be used to customize behavior when the element gets or loses the focus or when the Listbox is clicked.

Actions

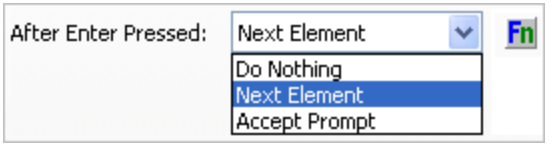
After Scanning

When a user scans a response for the Listbox element (as long as scanning is configured as a valid input source on the Input tab), the After Scanning setting determines how the element will behave. There are 3 choices. The default behavior is for the element to do nothing after the scan. You can choose to have the element advance the focus to the next element on the prompt. The order of the elements is determined on the prompt's settings screen. The last choice is to have the scan act as the acceptance for the entire prompt and thus the program advances to the next prompt as defined on the prompt's settings screen.



After Enter Pressed

When a user enters a response for the Listbox element, it is often desirable to have the data collection program automatically advance to the next element so that the user does not have to click on the next element to keep entering data. To help minimize the number of clicks required for the user, it is often a good idea to take advantage of the After Enter Pressed setting. There are 3 choices for how the element will behave after the Enter button is pressed. The default behavior is for the element to do nothing. You can choose to have the element advance the focus to the next element on the prompt. The order of the elements is determined on the prompt's settings screen. The last choice is to have the Enter key act as the acceptance for the entire prompt and thus the program advances to the next prompt as defined on the prompt's settings screen.



Events

The List Box element has several special event-driven scripts that allow for customized behavior.

- The **OnGetFocus** script runs when the element receives the focus. This can happen if the user clicks on the element, tabs to the element, if the focus is set to the element in a script, or can occur when a prompt is displayed if the element is the first element on the prompt.
- The **OnLoseFocus** Script runs when the element loses the focus, or in other words, when the focus is moved to another element.
- The **OnClick** Script runs whenever the element is tapped or clicked.
- The **OnDoubleClick** Script runs whenever the element is double-tapped or double-clicked.
- The **ValueChanged** Script runs whenever the data in the control is changed, whether by scanning, data entry, or assigning from a script.
- The **Validation** Script runs when the prompt is accepted, as part of the prompt validation process. If you call the *ValidationFail* function from within this script, the prompt validation will fail, the error message specified will be displayed, and the focus will be set back to this element.

Multilist Element

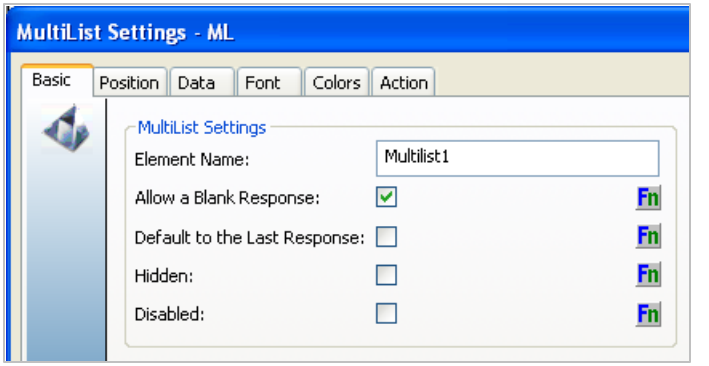
The Multilist element allows multiple items to be selected at the same time. It is similar to the Listbox, but the Listbox allows only one item to be selected. Each item in the Multilist corresponds to a piece of collected data whereas the entire Listbox corresponds to a piece of collected data.

To add a Multilist element to a prompt, click on the Multilist 

from the element list on the right side of the main Generator Application screen. If you cannot see the element list, you can turn it on by checking the Elements menu under the View main menu item. After you click the Multilist element, the Multilist Settings screen will be displayed. You will need to specify the settings for your Multilist. The key settings are the element Name and the data in the Multilist. These and all Multilist Settings are discussed below.

Basic Tab

The Basic tab on the Multilist Settings screen is shown here.



Element Name

Multilist elements, like all elements, must have a name. When you first create the Multilist element the name will be displayed as “Multilist1”. You change the name to any valid name. The name of the Multilist element can be up to 20 characters in length and must be a unique name within the prompt. The following characters may not be used on Input Text element names: # $ @ _ - * . / = < > ( ) & as they are reserved.

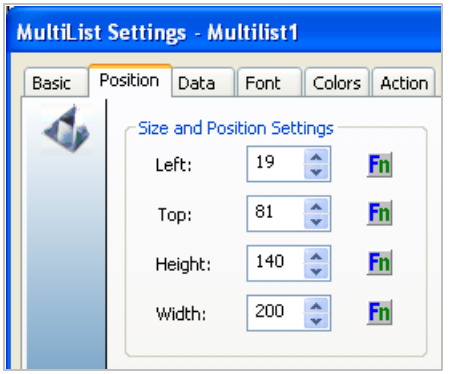
Allow a Blank Response

This setting will determine whether or not to allow a blank response for the Multilist element. By default this setting is turned on, meaning that a response is not required.

Hidden, Disabled

The Hidden setting will cause the Multilist element and all its items to be hidden when checked. The Disabled setting will cause the Multilist element to be disabled. A response cannot be entered, meaning the Multilist options cannot be clicked or scrolled, if the Multilist element is hidden and/or disabled.

Position Tab



The Position Tab on the Multilist Settings screen controls the size and position of the Multilist element on the prompt. You can also control the size and position of the Multilist element from the main design window. You can move the Multilist by selecting it and holding the mouse while you move the Multilist to the desired location. You can also resize the Multilist by selecting it and hovering over the resize boundaries of the Multilist element and dragging the Multilist to resize it.

Left Position

The Left Position setting specifies the horizontal location in pixels of the upper-left corner of the Multilist. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Left Position setting is set to 0, the Multilist will be all the way to the left of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the left position to a value larger than the screen the element will be located off the prompt screen.

Top Position

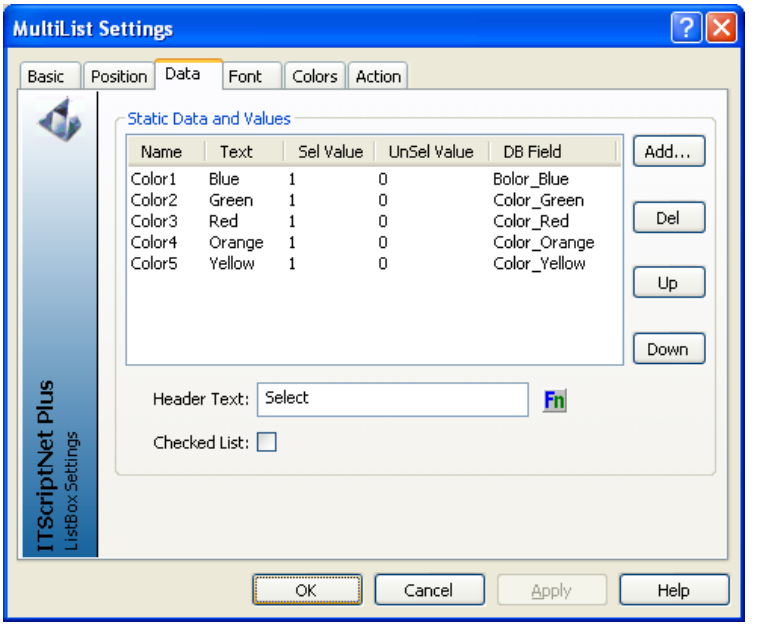
The Top Position setting specifies the vertical location in pixels of the upper-left corner of the Multilist. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Top Position setting is set to 0, the Multilist will be all the way to the top of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the top position to a value larger than the screen the element will be located off the prompt screen.

Height & Width

The Height setting specifies the height, in pixels, of the Multilist element. The Width setting specifies the width in pixels of the Multilist.

Data Tab

This tab is where you specify the Multilist items. Multilist items must be defined as static elements; however, you can use the settings in-prompt scripts to modify the settings during run-time in order to make these 'static' data items somewhat dynamic. Each item in the list of items for the Multilist must be entered into the Data tab.



Add

Click the Add… button to bring up the Edit Element box. For each Multilist item, you must define several settings.

Del

The Delete button removes the selected Multilist item from the list.

Up/Down

The Up and Down buttons allow you to manipulate the order of the list of Multilist choices listed. The Up and Down buttons move the selected Multilist item either Up or Down. Double-clicking the item in the Static Data and Values list will bring up the Edit Element screen so that each setting for the item can be edited.

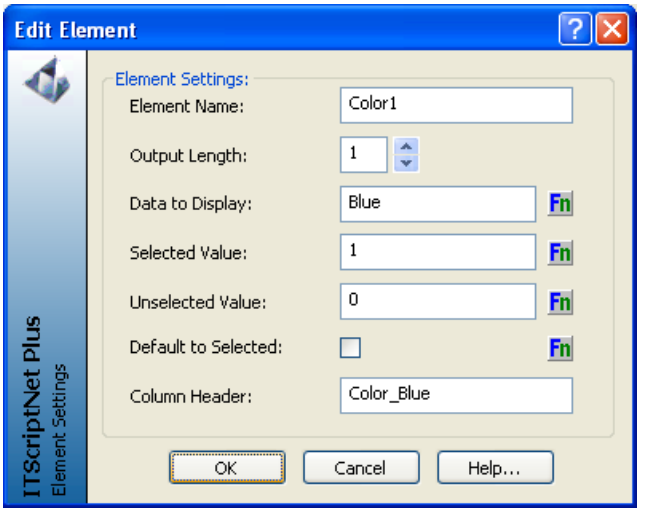
Header Text

The Header Text setting will set the title, or header, of the Multilist element.

Checked List

When the Checked List setting is checked, the Multilist will include checkboxes for each item in the list. The checkboxes may provide a more user-friendly multi-select list and users will find this style of Multilist very intuitive. A Multilist without the Checked List setting has its items selected by clicking to select the item or items. Shift-Click will allow multiple items to be selected.

Add/Edit Elements



The Edit Elements screen contains the settings for each item in a Multilist.

Element Name

This is the name of the Multilist's item. In order to reference an item of a Multilist, you will need to use the element name that you provide on this screen for the item. The same naming restrictions apply to the Multilist items as to the names of all elements, # $ @ _ - * . / = < > ( ) & as they are reserved.

Output Length

The Output Length is the collected data size for the Multilist item. Each item in a Multilist will collect data, so each item needs to have a defined size for the collected data.

Data to Display

The data to display is the display text that you want your user to see in the list for the item.

Selected Value

If the item is one of the items selected by the user, the data stored for the item will be the value defined in the Selected Value setting.

Unselected Value

If the item is not one of the items selected by the user, the data stored for the item will be the value defined in the Unselected Value setting.

Default to Selected

If the Default to Selected setting is checked, the item in the Multilist will default to be one of the selected items for the Multilist.

Database Field/Column Header/Field Header

This setting specifies where the data collected for this element is stored after it is sent to the PC and processed. Depending on the settings specified in the *Configure Receive* screen, this field may have different meanings.

If the program is configured to send its data to a Microsoft Access file or an ODBC data source, the target database field must be set up properly so the data collected can be sent to the desired field. A field from the target database must be selected from the list of fields. The fields in the list are the fields defined for the database and table that are established on the *Configure Receive* screen. The <Ignore> selection can be chosen and the collected data for the element will be ignored when the collected data is sent to the database. Fields are output to

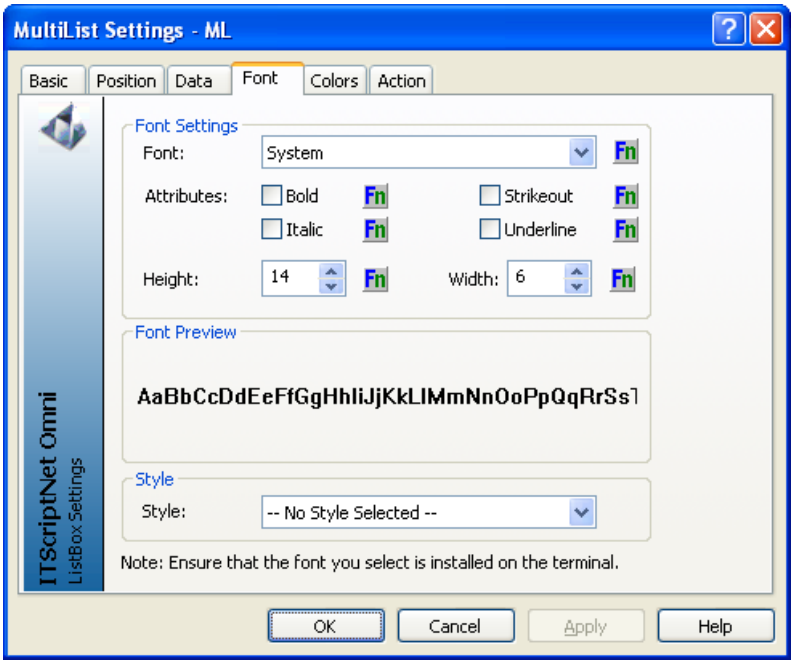
Access and ODBC if the field specified here exists and the maximum length of the response is not zero.

If the program is configured to send its data to a Microsoft Excel file, this setting will become the column header for the data. For Excel files, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen.

If the program is configured to send its data to a text file with a header, this setting will become the field header for the data in the output file. For text files with headers, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen. If the program is configured to send its data to a text file with no header information, then there will be no Field Header setting on the Multilist Settings screen.

Font Tab

The Font Tab on the Multilist Settings screen allows you to customize standard Multilist elements.



Font

The Font selection allows you to choose a font for the Multilist. Note that if you select a font that is not installed on your portable terminal, the operating system of the terminal will convert the font to a font that is installed. The effect of this conversion will most likely be that the positioning of the font for the Multilist on the terminal will not be consistent with the design view. The area under the font selection displays a preview of the font you have chosen.

Attributes

The Attributes section allows you to include Bold, Italic, Underline, or Strikeout attributes to your font choices. A check next to the attribute turns that attribute on.

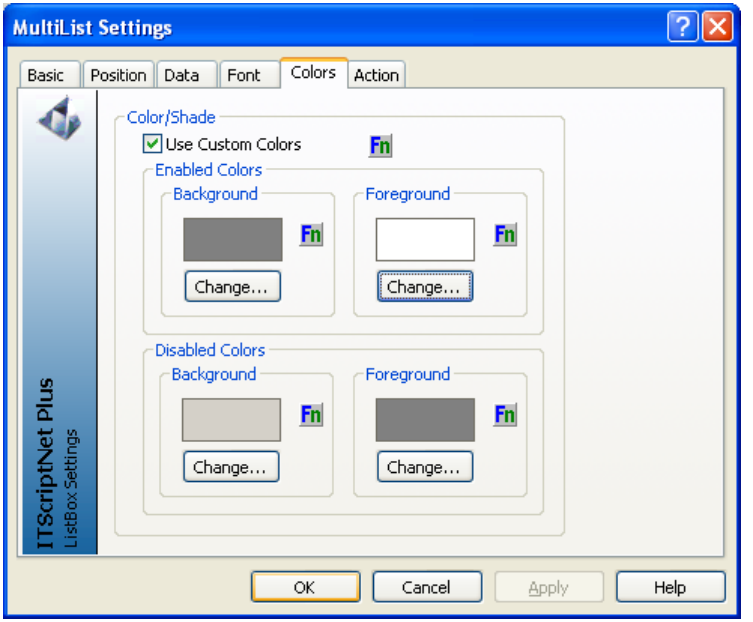
Height & Width

The Height and Width font settings specify the Height and Width of the font size in pixels. The Font Preview section will reflect the Height and Width specified. If you use a Font Width of zero, Windows will select an appropriate default width.

Style

The Style setting allows you to apply a pre-defined Font and Color set to this element. The Style must have been created with the Style Editor on the Program menu. Any changes made to the Style in the Style Editor will automatically be applied to this element. If you change any Font or Color setting on the element, the Style will be removed.

Colors Tab



The Colors tab adds another dimension of flexibility and customization to the design of Multilist elements. The colors will only be effective for portable terminals that have a color display. The first step towards customizing the colors of a standard button is to check the 'Use Custom Colors' option on the Colors tab. Once checked, the colors specified on the tab will be used.

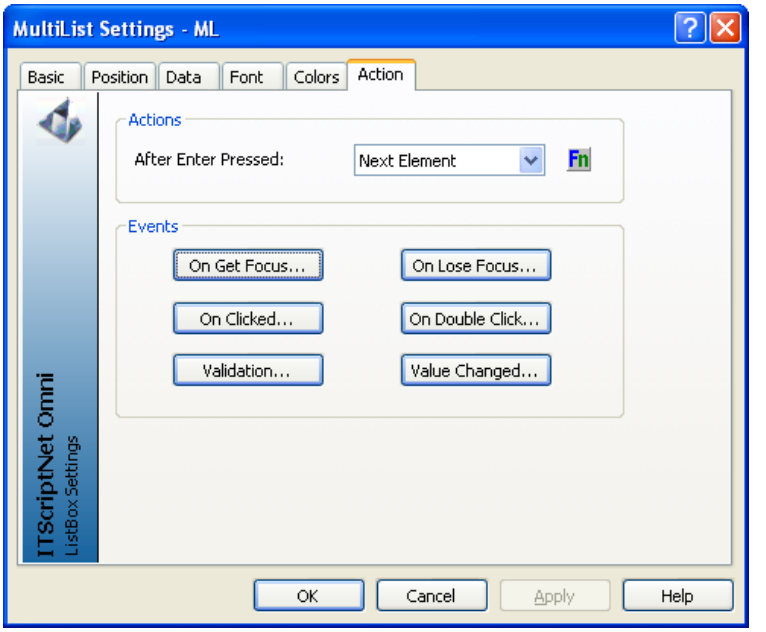
Enabled Colors

The Background color and the Foreground (Text) color on the Text Input can be specified independently. The enabled colors are used when the element is enabled. Click on the Change… button to bring up the Color screen. You can select from the basic colors on the left, select a color from the color matrix on the right, specify Hue/Saturation/Luminosity, or specify Red, Blue, and Green values to specify a custom color.

Disabled Colors

The Background color and the Foreground (Text) color on the buttons can be specified independently. The disabled colors are used when the button is disabled and not able to be clicked. Depending on the terminal, the disabled colors may be overridden by the operating system of the terminal and the disabled colors may not be displayed as designed.

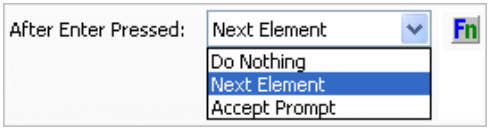
Action Tab



The Action tab contains the settings to control the behavior of the Multilist element after a response has been scanned and/or if the user presses the Enter button. This tab also has access to the in-prompt scripts that can be used to customize behavior when the element gets or loses the focus.

After Enter Pressed

When a user enters a response for the Multilist element, you may want the data collection program to automatically advance to the next element. To help minimize the number of clicks required for the user, it is often a good idea to take advantage of the After Enter Pressed setting. There are 3 choices for how the element will behave after the Enter button is pressed. The default behavior is for the element to do nothing. You can choose to have the element advance the focus to the next element on the prompt. The order of the elements is determined on the prompt's settings screen. The last choice is to have the Enter key act as the acceptance for the entire prompt and thus the program advances to the next prompt as defined on the prompt's settings screen.



Event

The Multilist element has several special event-driven scripts that allow for customized behavior.

- The **OnGetFocus** script runs when the element receives the focus. This can happen if the user clicks on the element, tabs to the element, if the focus is set to the element in a script, or can occur when a prompt is displayed if the element is the first element on the prompt.
- The **OnLoseFocus** Script runs when the element loses the focus, or in other words, when the focus is moved to another element.
- The **OnClick** Script runs whenever the element is tapped or clicked.
- The **OnDoubleClick** Script runs whenever the element is double-tapped or double-clicked.
- The **ValueChanged** Script runs whenever the data in the control is changed, whether by scanning, data entry, or assigning from a script.
- The **Validation** Script runs when the prompt is accepted, as part of the prompt validation process. If you call the *ValidationFail* function from within this script, the prompt validation will fail, the error message will be displayed, and the focus will be set back to this element.

Grid Element

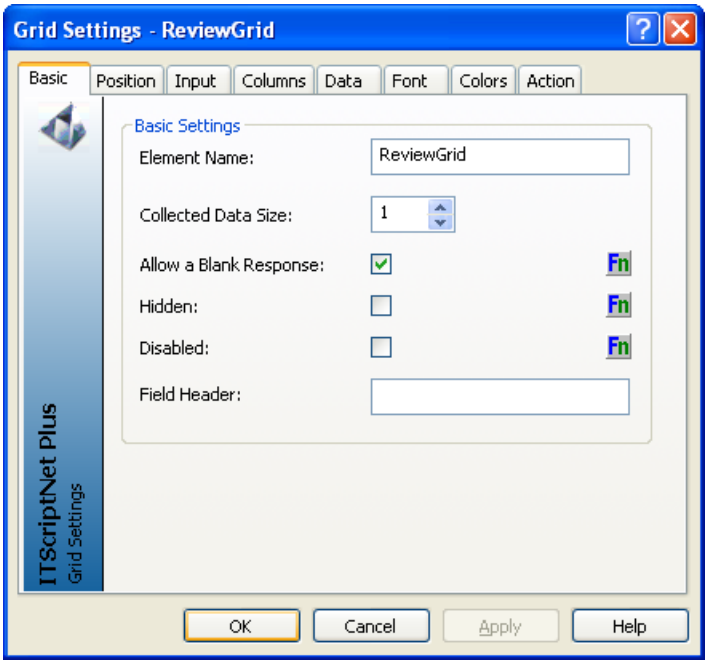
The Grid element presents a list of items to the user similar to the ListBox. However, the Grid element contains multiple pieces of information that can be displayed. Each type of information is in a column in the grid and each item consists of one row in the grid. One of the aspects of the Grid element that makes it so powerful is that it supports displaying information from static content, from a Validation file, or from collected data.

To add a Grid element to a prompt, click on the Grid 

element from the element list on the right side of the main Generator Application screen. If you cannot see the element list, you can turn it on by checking the Elements menu under the View main menu item. After you click the Grid element, the Grid Settings screen will be displayed. You will need to specify the settings for your Grid. The key settings are the element Name, the collected data size, and whether the data for the Grid will be pre-defined (static), come from a validation file, or come from collected data. These and all Grid Settings are discussed below.

Basic Tab

The Basic tab on the Grid Settings screen is shown here.



Element Name

Grid elements, like all elements, must have a name. When you first create the Grid element the name will be "Grid1". You may change the name to another valid element name. The name of the Grid element can be up to 20 characters in length and must be a unique name within the prompt. The following characters may not be used on Input Text element names: # $ @ _ - * . / = < > ( ) & as they are reserved.

Collected Data Size

The Collected Data Size setting is the size of the underlying values that are associated with the Item Data of the grid. The Item Data is the piece of data that is stored when a row in the grid is selected for data collection. The item data may or may not be included in the grid as a column.

Allow a Blank Response

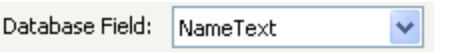
This setting will determine whether or not to allow a blank response for the Grid element. By default this setting is turned on, meaning that a response is not required. If the box is checked, thus allowing a blank response, the user collecting the data will be able to go to the next prompt even if he has not chosen one of the Grid items and therefore the Grid element has no response and is blank. Allowing a blank response is useful for optional data in a data collection program. There may be times when the grid is used as a means to display information to a user and therefore would not need to have data collected for the Grid element itself.

Hidden, Disabled

The Hidden setting will cause the Grid element and all its items to be hidden when checked. The Disabled setting will cause the Grid element to be disabled. A response can not be entered, meaning the Grid options can not be clicked or scrolled, if the Grid element is hidden and/or disabled.

Database Field/Column Header/Field Header

This setting specifies where the data collected for this element is stored after it is sent to the PC and processed. The data collected for a Grid element will be the Item Data associated with the selected row. Depending on the settings specified in the *Configure Receive* screen, this field may have different meanings.

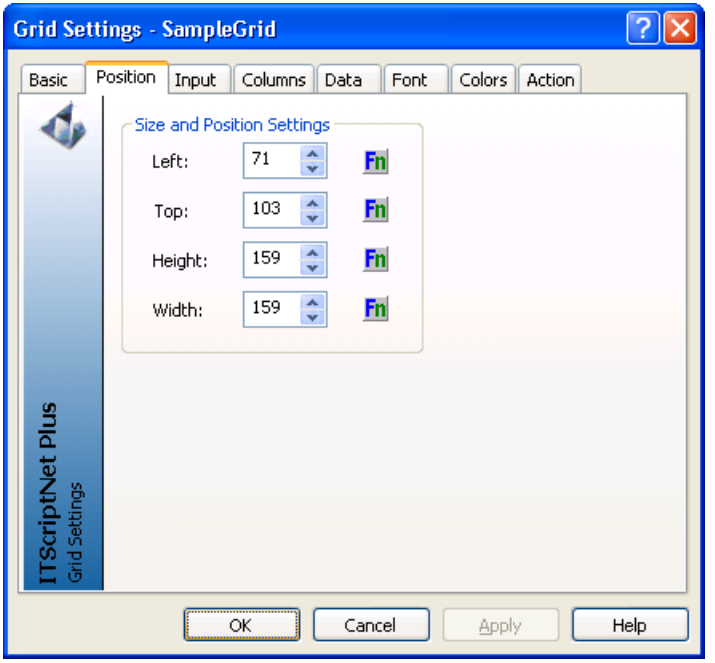


If the program is configured to send its data to a Microsoft Access file or an ODBC data source, the target database field must be set up properly so the data collected can be sent to the desired field. A field from the target database must be selected from the list of fields. The fields in the list are the fields defined for the database and table that are established on the *Configure Receive* screen. The <Ignore> selection can be chosen and the collected data for the element will be ignored when the collected data is sent to the database. Fields are output to Access and ODBC if the field specified here exists and the maximum length of the response is not zero.

If the program is configured to send its data to a Microsoft Excel file, this setting will become the column header for the data. For Excel files, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen.

If the program is configured to send its data to a text file with a header, this setting will become the field header for the data in the output file. For text files with headers, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen. If the program is configured to send its data to a text file with no header information, then there will be no Field Header setting on the Grid Settings screen.

Position Tab



The Position tab on the Grid Settings screen controls the size and position of the Grid element on the prompt. You can also control the size and position of the Grid element from the main design window. You can move the Grid by selecting it and holding the mouse while you move the Grid to the desired location. You can also resize the Grid by selecting it and hovering over the resize boundaries of the Grid element and dragging the Grid to resize it.

Left Position

The Left Position setting specifies the horizontal location in pixels of the upper-left corner of the Grid. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Left Position setting is set to 0, the Grid will be all the way to the left of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the left position to a value larger than the screen the element will be located off the prompt screen.

Top Position

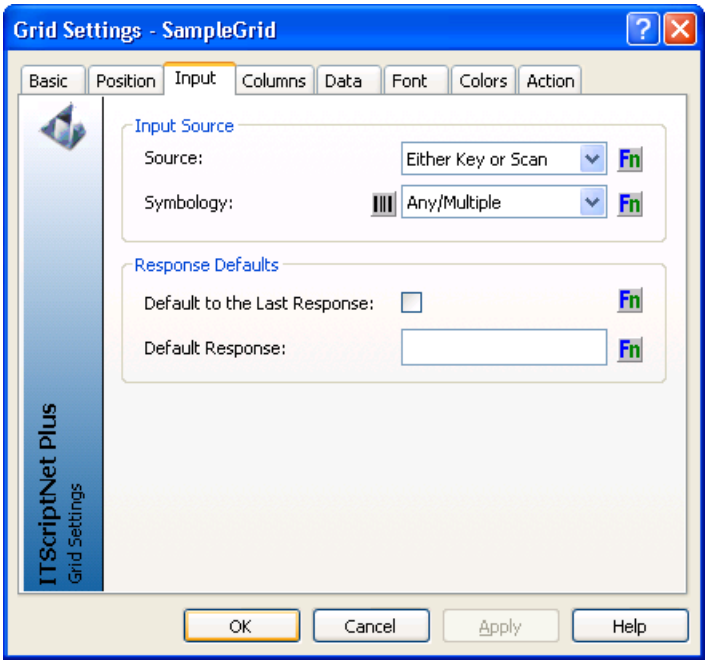
The Top Position setting specifies the vertical location in pixels of the upper-left corner of the Grid. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Top Position Setting is set to 0, the Grid will be all the way to the top of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the top position to a value larger than the screen the element will be located off the prompt screen.

Height & Width

The Height setting specifies the height, in pixels, of the Grid element. The Width setting specifies the width in pixels of the Grid.

Input Tab

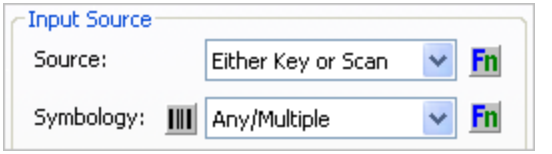
Input Source



The setting for Source specifies whether the response to the Grid can be entered only on the terminal's keypad, only by scanning a barcode, or by either keypad or scan. By default, Grids are set to allow either keypad or scan for data entry. Most likely users will select an item from the Grid using the arrow keys or by tapping on the screen.

Symbology

The setting for Symbology is used to limit the acceptable barcode symbologies when scanning a response for a Grid element. There are many different barcode symbologies. A barcode symbology is an algorithm for encoding data into a barcode. Some of the more common symbologies are: Code 39, I2of5, Code 128, and UPCA. The Grid element can be set to accept any barcode symbology (the terminal has to be able to decode the symbology, please refer to documentation on the specific portable terminal you are using for a list of symbologies compatible with your hardware). The Any/Multiple selection will allow a user to scan any symbology that is configured as a default for the scanner/imager for your hardware. The Any/Multiple setting is the default for any new Grid elements that you create.



The Symbology property can be set to accept only one type of barcode symbology. To select a single symbology, select it from the list. The default settings for the symbology on your terminal will be used. By way of example, let's say that your warehouse uses Code 39 barcodes for your part numbers. However, the boxes also contain UPC codes for those products. When collecting data, you want to insure that the Code 39 barcode is scanned and not the UPC code. By setting the Grid element to only accept Code 39, the program will reject the UPC codes and any other symbology. Specifying a specific symbology is a means of validating your data and *maximizing data accuracy.*

Depending on your data collection needs, you may need to customize the Symbology setting. In our example above, the warehouse had Code 39 and UPC part numbers. Let's say that now you need to be able to scan either a Code 39 or a UPC, but still do not want to allow any other symbologies. You can do this by clicking on the button with the barcode icon to bring up the *Advanced Symbology Settings* screen. Please refer to the section in this User Guide describing the *Advanced Symbology Settings* screen. The *Advanced Symbology Settings* screen will also allow you to control each parameter available for one or more symbologies. If you do not customize the barcode symbology settings, the default settings for the single symbology specified or all symbologies will be used. The defaults are appropriate for most data collection applications, but if you need to control the settings more precisely, **IT*Script*Net** allows you access and full control of the symbologies. If the *Advanced Symbology Settings* screen is used to customize the symbology behavior for a Grid, the barcode icon will change colors to provide an easy visual indicator that customized symbology settings are in use.

Response Defaults

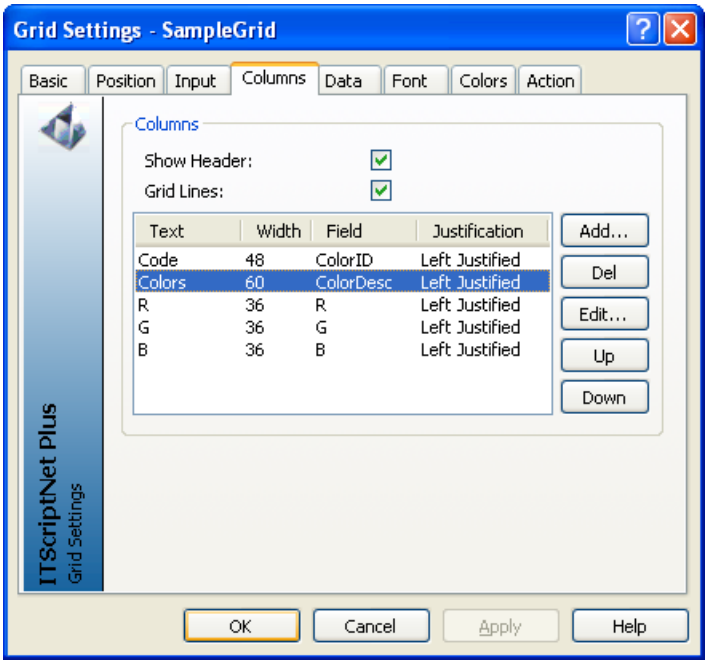
The Response Defaults settings can set the default response for the element to a specific value or the value of the response from the last data record collected for the element. When creating a new Grid element, the response default is initially set to no default. When in this state, the user will have to enter data by selecting an item from the Grid each time through the prompting loop during data collection.

Using response defaults, the *Default to the Last Response* option can be enabled and will cause the response to default to the previous response. The user will still be able to change the response, but can save time by accepting the default. This feature is useful when an element will often have the same information for several prompting loops.

The other option available for the response default is to actually set a response default that will always be shown as the default response for the Grid. It is important to note that the value of the default response specified must match one of the item data fields in the Grid – not necessarily any of the text that is shown in the Grid.

Columns Tab

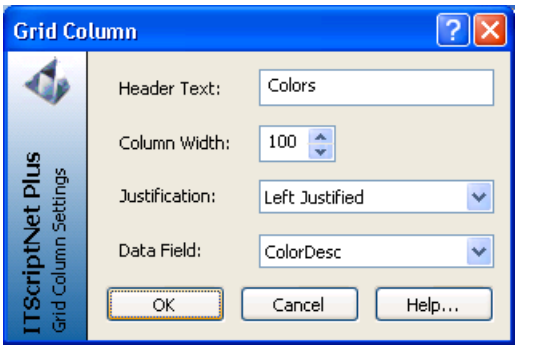
The Columns tab defines the data to display in the grid. The settings on the Columns tab work in conjunction with the settings on the Data tab. The Data tab allows you to choose whether the grid will be filled from static data, a validation file, or from collected data. No matter what option is used for the grid data, the columns to display the data must be defined on the Columns tab.



Show Header and Grid Lines

The Show Header and Grid Lines settings control the look of the grid. If the Show Header option is checked, then the Header Text for each column will be displayed in the Grid Header. If the Show Header option is not checked, the grid will not have a header and will only display its data. The Grid Lines setting determines whether or not the grid will contain grid lines. The grid lines separate each cell in the grid with lines. These options can customize the look of your grid element, but it is not recommended that both options be used because they will conflict with each other when displayed.

Add/Edit/Delete Grid Column



You can Add a Grid Column by clicking the Add… button. This will bring up the Grid Column screen. The Grid Column screen defines a column. The Header Text is the text that will appear as the heading for the column if the Show Header setting is enabled for the grid element. The Column Width is the width of the column in pixels. The Justification allows you to select whether the information in the columns should be displayed left, right, or center justified. The Data Field column links one of the fields in the data source to the column. The fields to choose from for the data field will depend on the data source specified in the Data tab. To Edit an existing column, double-click a column in the list or select the column to edit and click the Edit… button. Use the Del button to remove a column from the grid.

Up/Down

You can change the relative positions of the columns you defined using the Up and Down buttons.

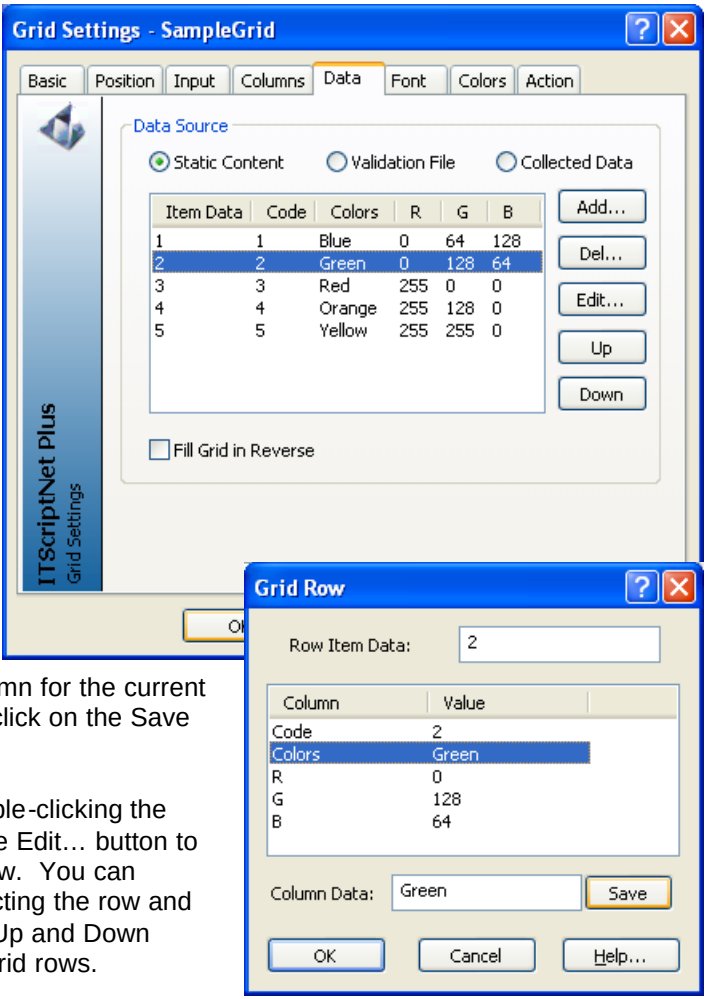
Data Tab

This tab is where you specify the source for the data that is to be displayed in the Grid. There are three types of data content sources for grids: Static, Validation File, and Collected Data. Each of these data source options is defined in this section.

One common setting amongst all the data source options is the Fill Grid in Reverse option. When this box is checked the data will fill the grid in reverse order. This may be desired for situations where it is useful to have the most recent item displayed in the grid first instead of last.

Static Content Grid Data Source

For Static Content, each piece of data in the grid must be entered into the Data tab. For Static Content, you will want to define your columns on the Columns tab first. Once your columns are defined, you can then enter the data for the Grid.



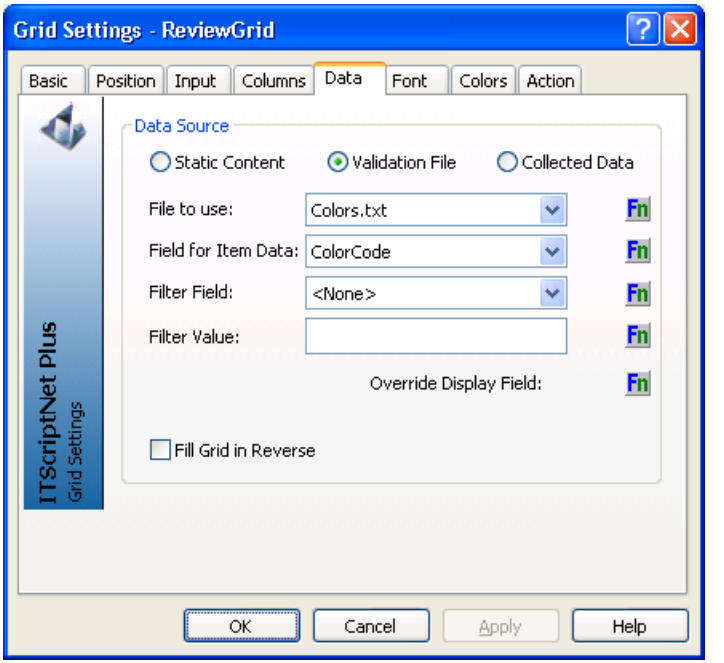
Click the Add… button to add a row to the grid. The Grid Row screen will allow you to specify data for each column for a given row of data to be displayed.

The Row Item Data will contain the data that will be stored for the grid if the current row is selected by the user in data collection. To enter data for each column, select the column from the list, enter the data for that column for the current row in the Column Data field, and then click on the Save button to update the column data.

You can edit any row in the grid by double-clicking the row on the Data tab list or by clicking the Edit… button to bring up the Grid Row screen for that row. You can delete a row of data for the grid by selecting the row and then clicking the Del…button. Use the Up and Down buttons to change the positions of the grid rows.

Validation File Grid Data Source

Validation files can be used to fill a Grid element dynamically at run-time instead of at design time. This gives your data collection program more flexibility. When using a validation file as the data source for the grid, you will want to set-up the validation file on the Data tab first and then go to the Columns tab to define the columns and link them with the validation file fields.

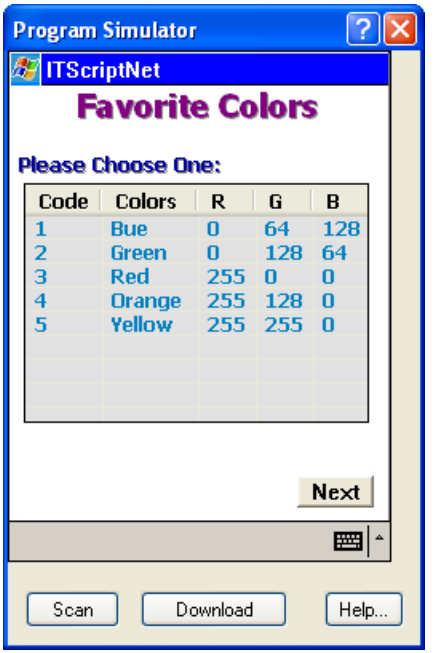


The 'File to use' drop-down box allows you to select a validation file to use to fill the Grid. The validation text file must be defined from the *Validation Files* screen before a Grid can be configured to use the validation text file. Please refer to the *Validation Files* Screen section of this User Guide for more information on setting up the validation file for your data collection program.

Once the validation file has been selected, the Field for Item Data drop-down list will be filled with the fields in the validation file. Select a field to use as the Item Data. The Item Data is the underlying data in the Grid that will be stored for the item if the item is chosen by the user when collecting data. The Item Data is independent from the data displayed in the columns. The validation file field that you choose for the Item Data may also be used in a column, but it is not required to be displayed.

The *Filter Field* is used to apply a filter to the validation file so that only matching records are added to the picklist. For example, if you were filling a Picklist from an Order Items file, you might specify an Order Number field as the filter field, and a specific order number as the Filter Value so that only items on that order are listed in the Picklist. The *Filter Value* is the value that should be used to filter the validation file. You can enter a value, or use the In-Prompt Script to override the Filter Value so you can filter by the value of another prompt or variable.

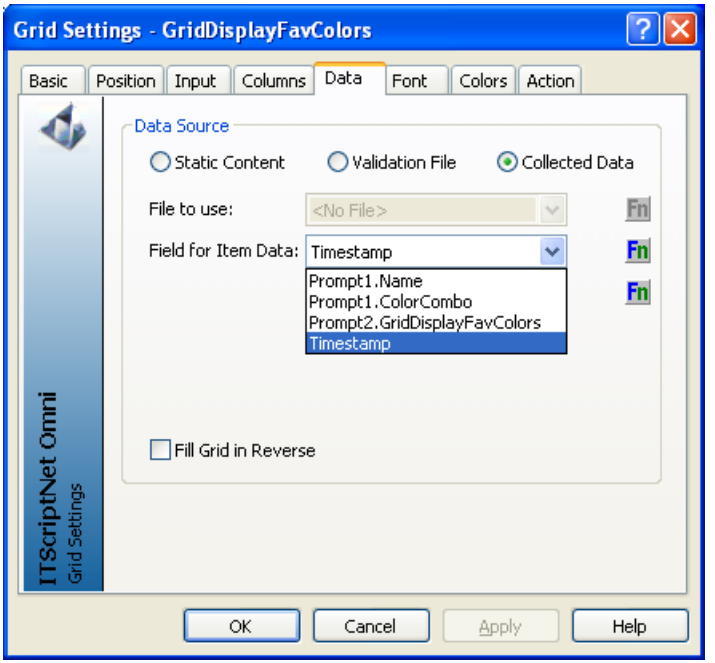
The *Override Display Field* allows you to control the way data is presented in the grid even further than the Filter and Display options already described. This script is executed once for each record in the validation file. If the script returns 0, the record will be added to the grid. If the script returns non-zero, the record will not be added to the grid. You can reference the fields in the validation file record using the *PickListField* function. You can override the data to be added to a column by assigning using the format $prompt.grid.column$ = value.



The example shown uses a validation file called *colors.txt*. There is a field called ColorID, which contains a numeric code for each color listed. The names of the colors are in the field named ColorDesc and will be displayed to the user in the Grid. This example provides exactly the same results as shown in the Static Content section.

Collected Data Grid Data Source

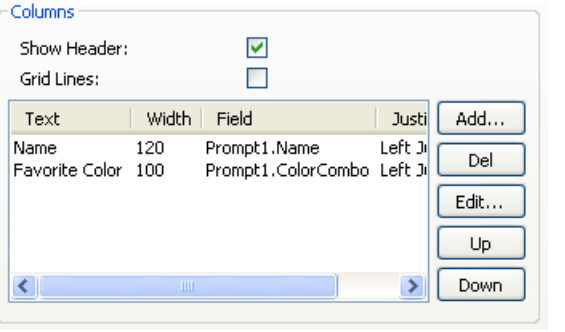
In addition to having static content or having data from a validation file, Grids can also be filled from collected data. This is another way to fill a Grid element dynamically at run-time instead of at design time. Using collected data as the data source for a grid is very similar to using a validation file to fill the grid. When using collected data as the data source for the grid, you will want to set-up the Data tab first and then go to the Columns tab to define the columns and link the columns with the data collection fields.



The 'File to use' drop-down box will be disabled for collected data, since there is no file choice—there is only one set of collected data for each data collection program! The Field for Item Data drop-down list will be filled with the fields in the collected data. Each field in the list corresponds to a data collection prompt or element in your program. Select a field to use as the underlying data in the Grid that will be stored for the item if the item is chosen by the user when collecting data. In this case, the timestamp was selected.

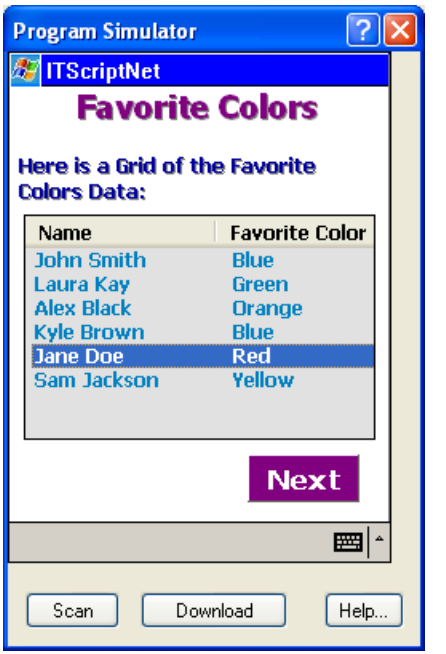
Although the timestamp does not correspond to an element or prompt in the program, the timestamp is always collected with each record. Choosing the timestamp was a convenient choice because it will be unique with the collected data.

The Columns tab and the Data tab need to be defined to work together. The figure shows a portion of the Columns tab for this grid. The header text and width is the same as for the static and validation grids. Note, however, that the Field for each column is the name of a data input element. To refer to a data input element, use the Prompt.Element convention.



The example shows the simulator view of a program that collects a name and a favorite color on the first prompt in the program. Then, on the second prompt, the data that has been collected is displayed in the grid. There is a Name column that displays the name entered and there is a 'Favorite Color' column that displays the color stored for that person.

When using a grid to display collected data, make sure that you save the data by creating a looping structure that allows the user to get to the last prompt to save the data before you display the grid. Otherwise, the data for the current loop will not be displayed since it will not have been saved yet!

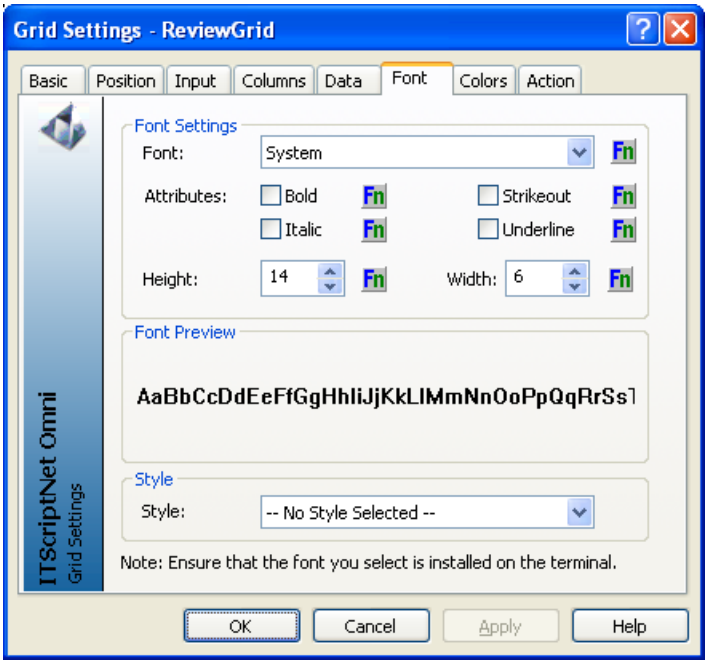


Font Tab

The Font Tab on the Grid Settings screen allows you to customize standard Grid elements.

Font

The Font selection allows you to choose a font for the Grid. Note that if you select a font that is not installed on your portable terminal, the operating system of the terminal will convert the font to a font that is installed. The effect of this conversion will most likely be that the positioning of the font for the Grid on the terminal will not be consistent with the design view. The area under the font selection displays a preview of the font you have chosen.



Attributes

The Attributes section allows you to include Bold, Italic, Underline, or Strikeout attributes to your font choices. A check next to the attribute turns that attribute on.

Height & Width

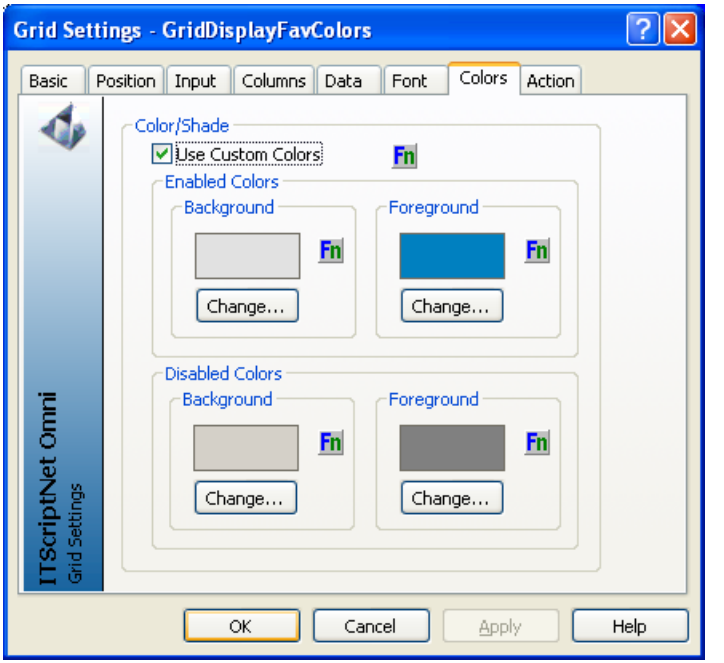
The Height and Width font settings specify the Height and Width of the font size in pixels. The Font Preview section will reflect the Height and Width specified. If you use a Font Width of zero, Windows will select an appropriate default width.

Style

The Style setting allows you to apply a pre-defined Font and Color set to this element. The Style must have been created with the Style Editor on the Program menu. Any changes made to the Style in the Style Editor will automatically be applied to this element. If you change any Font or Color setting on the element, the Style will be removed.

Colors Tab

The Colors tab adds another dimension of flexibility and customization to the design of Grid elements. The colors will only be effective for portable terminals that have a color display. Check the 'Use Custom Colors' option on the Colors tab to enable the colors. Once checked, the colors specified on the tab will be used.



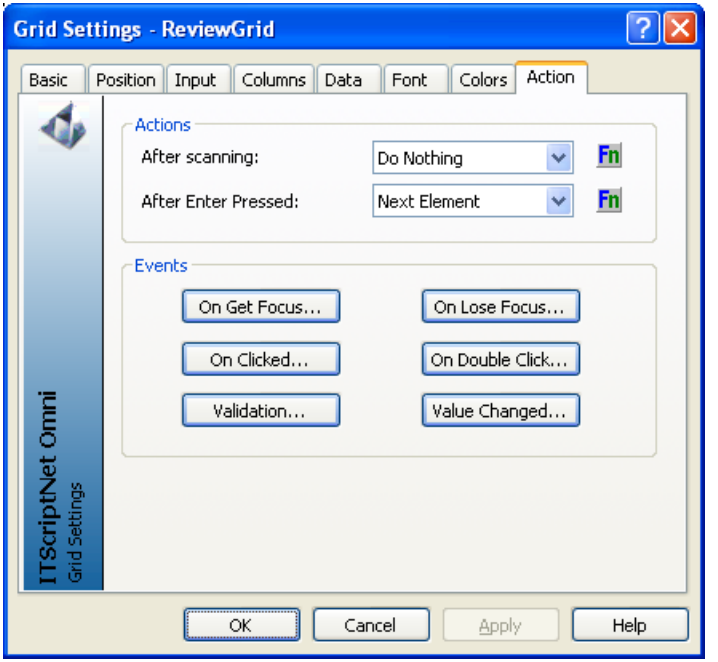
Enabled Colors

The Background color and the Foreground (Text) color can be specified independently. The enabled colors are used when the element is enabled. Click on the Change… button to bring up the Color screen. You can select from the basic colors on the left, select a color from the color matrix on the right, specify Hue/Saturation/Luminosity, or specify Red, Blue, and Green values to specify a custom color.

Disabled Colors

The Background color and the Foreground (Text) color on the buttons can be specified independently. The disabled colors are used when the button is disabled and not able to be clicked. Note that Windows CE or PocketPC may override the disabled foreground color.

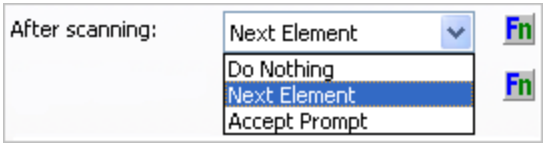
Action Tab



The Action tab contains the settings to control the behavior of the Grid element after a response has been scanned and/or if the user presses the Enter button. This tab also has access to the in-prompt scripts that can be used to customize behavior when the element gets or loses the focus or when the user clicks, or selects, a row of the grid.Actions

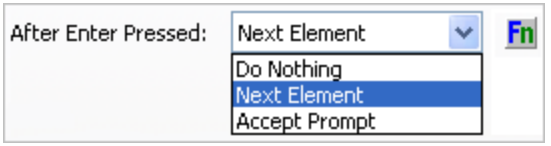
After Scanning

When a user scans a response for the Grid element (as long as scanning is configured as a valid input source on the Input tab), the After Scanning setting determines how the element will behave. There are 3 choices. The default behavior is for the element to do nothing after the scan. You can choose to have the element advance the focus to the next element on the prompt. The order of the elements is determined on the prompt's settings screen. The last choice is to have the scan act as the acceptance for the entire prompt and thus the program advances to the next prompt as defined on the prompt's settings screen.



After Enter Pressed

When a user enters a response for the Grid element, it is often desirable to have the data collection program automatically advance to the next element so that the user does not have to click on the next element to keep entering data. There are 3 choices for how the element will behave after the Enter button is pressed. The default behavior is for the element to do nothing. You can choose to have the element advance the focus to the next element on the prompt. The order of the elements is determined on the prompt's settings screen. The last choice is to have the Enter key act as the acceptance for the entire prompt and thus the program advances to the next prompt as defined on the prompt's settings screen.



Events

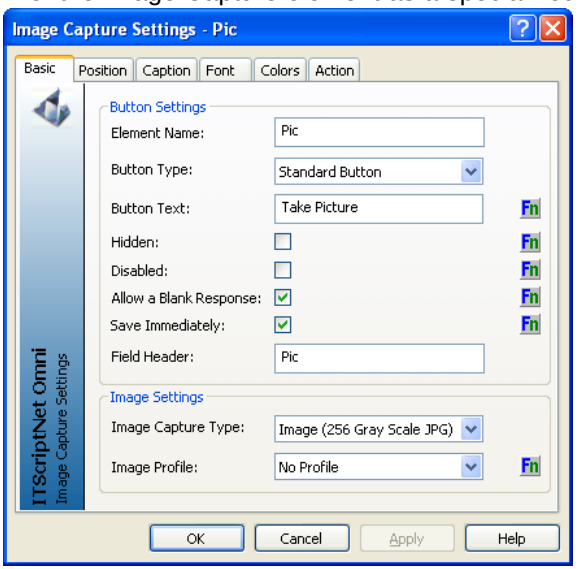
The Grid element has several special event-driven scripts that allow for customized behavior.

- The **OnGetFocus** script runs when the element receives the focus. This can happen if the user clicks on the element, tabs to the element, if the focus is set to the element in a script, or can occur when a prompt is displayed if the element is the first element on the prompt.
- The **OnLoseFocus** Script runs when the element loses the focus, or in other words, when the focus is moved to another element.
- The **OnClick** Script runs whenever the element is tapped or clicked.
- The **OnDoubleClick** Script runs whenever the element is double-tapped or double-clicked.
- The **ValueChanged** Script runs whenever the data in the control is changed, whether by scanning, data entry, or assigning from a script.
- The **Validation** Script runs when the prompt is accepted, as part of the prompt validation process. If you call the *ValidationFail* function from within this script, the prompt validation will fail, the error message specified will be displayed, and the focus will be set back to this element.

Image Capture Element

Image Capture element allows you to add a whole other dimension to your data collection programs by including pictures in your collected data. This element type is enabled for terminals with an **IT*Script*Net** supported imager only. The Image Capture element is very similar to the Action Button element. In fact, you can almost think of the Image Capture element as a specialized action button where the action is to take an image.

To add an Image Capture element to a prompt, click on the Image Cap element from the element list on the right side of the main Generator Application screen. If you cannot see the element list, you can turn it on by checking the Elements menu under the View main menu item. After you click the Image Capture element, the Image Capture Settings screen will be displayed. You will need to specify the settings for the Image Capture element. The key settings are the element Name and the Image Capture file type. These key settings and all the Image Capture Settings are discussed below.



Basic Tab

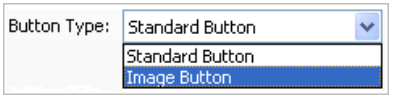
The Basic tab on the Image Capture Settings screen is shown here.

Element Name

Image Capture elements, like all elements, must have a name. When you first create the element the name will be displayed as “ImageCapture1”. You may change the element name to a valid element name. Element names can be up to 20 characters in length and must be unique within the prompt. The following characters may not be used on prompt names: # $ @ _ - * . / = < > ( ) & as they are reserved.

Button Type

The Image Capture element is a specialized Action Button. The Button Type setting is identical to the button type setting for Action Buttons. The Button Type can be either a Standard Button or an Image Button. The only difference between a standard button and an image button is that a standard button will display text and an image button will display the button with an image. Select the type of button that you want from the drop-down list.



Button Text

The Button Text setting is only applicable to Standard Buttons. The Button Text is the text that is displayed on the button. In the example shown, the Button Text setting is filled in with "Take Picture" in order to help provide useful instructions to the user of the data collection program.

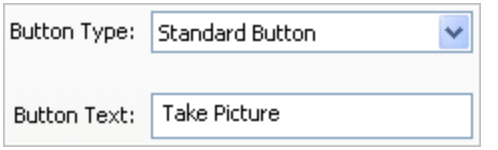
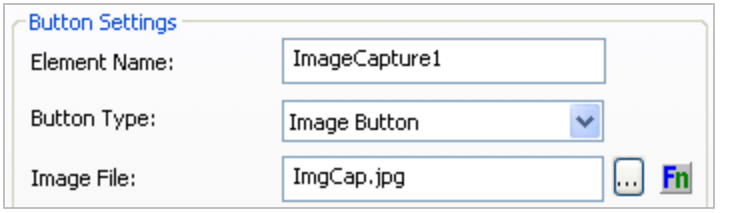


Image File

The Image File setting is only applicable to Image Buttons and not standard image capture buttons. The Image File is the filename of the graphic to be displayed on the button. Use the Browse button  to locate the graphic file for the Image Capture button image. **IT*Script*Net** supports bitmap (.bmp), jpeg (.jpg), and .png image file formats. The full path to the graphics file must be specified. If there is no path, as shown in the example, the graphic file must be located in the same directory as the data collection program file (.itb file).



The image file used for the Image Capture button will be automatically added to the Support Files list. Support files are used by the data collection program and would typically be sent to the terminal with the data collection program. Please refer to the section in the User Guide on Support Files for more information.

Hidden, Disabled

The Hidden setting will cause the image capture element to be hidden when checked. The Disabled setting will cause the element to be disabled. An Image Capture button that is hidden and/or disabled cannot be clicked and will not be able to capture an image.

Allow a Blank Response

The response for an image capture element is the image that was captured. The Allow a Blank Response setting determines if the image capture element is allowed to have a blank image as its response.

Save Immediately

This field controls whether the actual Image File collected for this Image Capture element is created immediately when the image is captured, or when the Collected Data Record is written to the file. By default, Images are saved Immediately.

Database Field/Column Header/Field Header

This setting specifies where the data collected for this element is stored after it is sent to the PC and processed. Depending on the settings specified in the *Configure Receive* screen, this field may have different meanings.

If the program is configured to send its data to a Microsoft Access file or an ODBC data source, the target database field must be set up properly so the data collected can be sent to the desired field. A field from the target database must be selected from the list of fields. The fields in the list are the fields defined for the database and table that are established on the *Configure Receive* screen. The <Ignore> selection can be chosen and the collected data for the Image Capture element will be ignored when the collected data is sent to the database. Fields are output to Access and ODBC if the field specified here exists. The name of the image will be stored if the field selected is a text field. If the field selected is an OLE object, then the actual image collected will be embedded in the database field.

If the program is configured to send its data to a Microsoft Excel file, this setting will become the column header for the data. For Excel files, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen. The file name of the image will be stored for Excel files.

If the program is configured to send its data to a text file with a header, this setting will become the field header for the data in the output file. For text files with headers, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen. If the program is configured to send its data to a text file with no header information, then there will be no Field Header setting on the Image Capture Settings screen. The image capture element will store the file name of the captured image as its data if the *Configure Receive* screen is set up for a text file.

Image Capture Type

The Image Capture Type Setting controls the format of the image that is captured. The two choices are 256-Color Gray Scale JPG or a 2-Color PNG that is often used for an image of a text document or a signature.

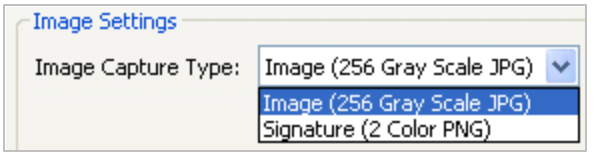
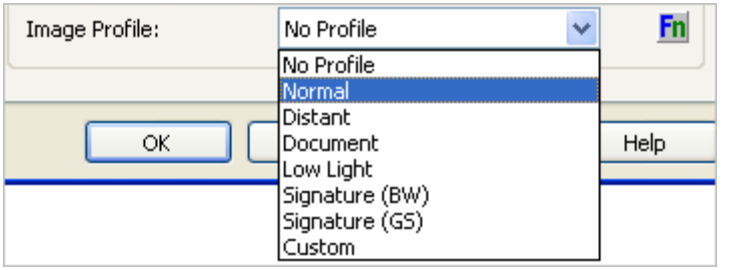


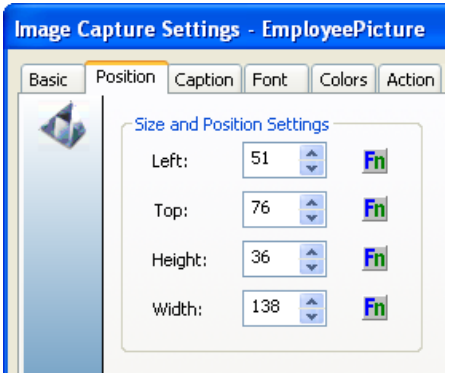
Image Profile

The Image Profile settings are only available for terminals equipped with an Imager that supports Image Profiles (see list on our web site). These imagers include the capability to apply processing to the image to optimize the images under certain conditions. The Image Profiles available are: Normal, Distant, Document, Low Light, Signature (Black & White), Signature (Grayscale), and Custom. The custom profile is typically configured in the imaging demo area on the terminals and is made available through **IT*Script*Net**. Refer to your terminal's documentation for more detailed information on image profiles.



Position Tab

The Position Tab on the Image CapturesSettings screen controls the size and position of the image capture button element on the prompt. You can also control the size and position of the image capture button from the main design window. You can move the image capture button element by selecting it and holding the mouse while you move the button to the desired location. You can also resize the element by selecting it and dragging the resize rectangles.



Left Position

The Left Position setting specifies the horizontal location in pixels of the upper-left corner of the Image Capture element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Left Position setting is set to 0, the image capture element will be all the way to the left of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the left position to a value larger than the screen the element will be located off the prompt screen.

Top Position

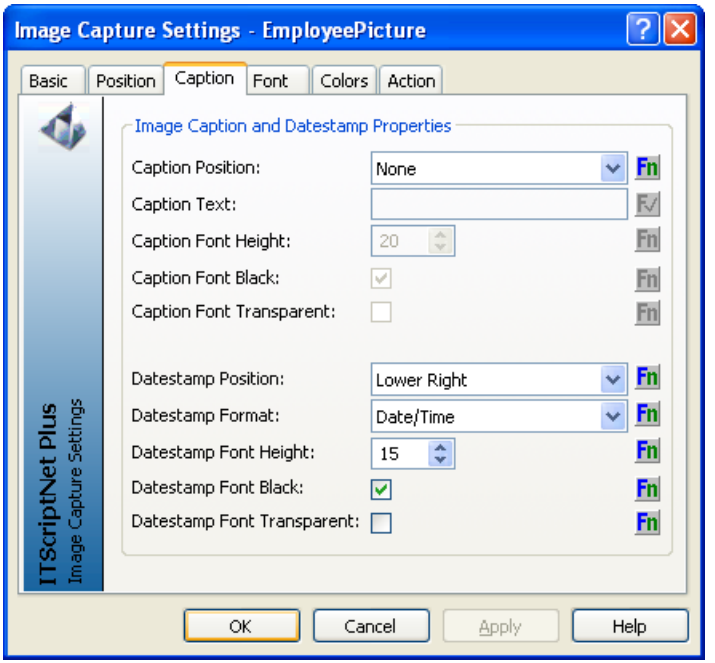
The Top Position setting specifies the vertical location in pixels of the upper-left corner of the Image Capture button. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Top Position setting is set to 0, the image capture button element will be all the way to the top of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the top position to a value larger than the screen the element will be located off the prompt screen.

Height and Width

The Height setting specifies the height in pixels of the Image Capture Button. The Width setting specifies the width in pixels of the Image Capture Button.

Caption Tab

Image Capture elements will collect image data. The Caption tab allows you to customize captions that can optionally be superimposed on the image. There are 2 possible pieces of information to include in the image. One piece of information is a general caption that you can specify. The other is a date/time stamp. You can choose to use neither, one, or both pieces of information to add detail to your captured images.



Caption Position

The Caption Position allows you to choose where in the image to include the caption text. If you choose "None", the caption will not be added to the image. The other choices are Upper Left, Upper Center, Upper Right, Lower Left, Lower Center, and Lower Right.

Caption Text

The Caption Text is the text that you want to be superimposed on the image you capture.

Caption Font

There are 3 settings related to the caption's font. The Caption Font Height is the number of pixels high the caption text will be. The Caption Font Black setting will make the caption text be black if checked, or white if not checked. The Caption Font Transparent setting will make the rectangle enclosing the caption text transparent if checked. If the Caption Font Transparent setting is not checked, the caption text will appear on a solid white or black rectangle, opposite the text color.

Datestamp Position

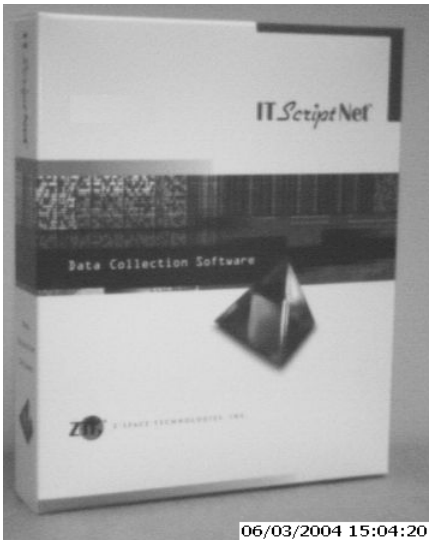
The Datestamp Position works the same as the Caption Position. You can choose to not include a Datestamp or you can choose the location on the image for the Datestamp to be written.

Datestamp Format

This setting allows you to choose the format for your date/time stamp. You can choose to include both the date and time, just the date, or just the time.

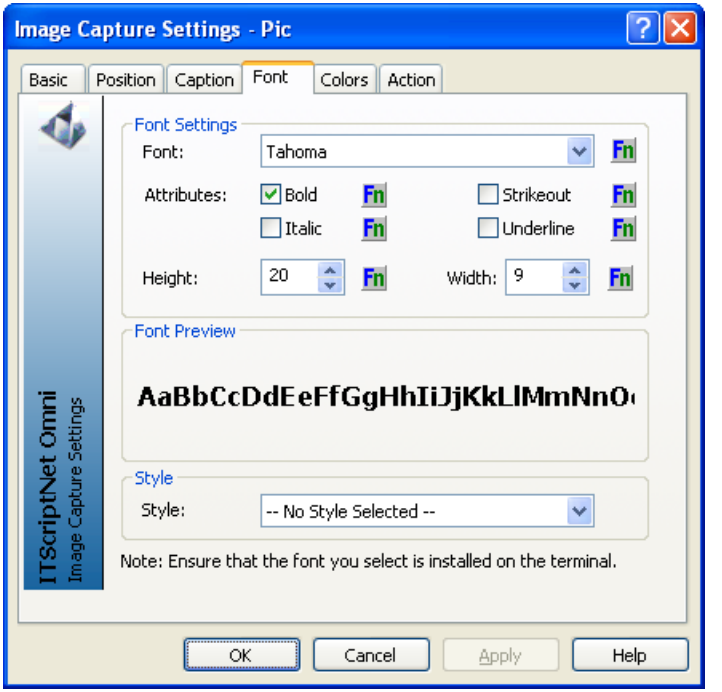
Datestamp Font

There are 3 settings related to the datestamp's font. The Datestamp Font Height is the number of pixels high the datestamp text will be. The Datestamp Font Black setting will make the datestamp text be black if checked or white if not checked. The Datestamp Font Transparent setting will make the rectangle enclosing the datestamp text transparent if checked. If the Datestamp Font Transparent setting is not checked, the datestamp text will appear on a solid white or black rectangle, opposite the text color.
The image shown at the right is an example of an image that was captured from an image capture element and enhanced with a Date/Time Stamp in the Lower Right corner. The Datestamp Font Black option is checked and the Datestamp Font Transparent option is not checked.



Font Tab

The Font tab on the Image Capture Settings screen allows you to customize standard Image Capture elements. The font settings will have no effect on image-style image capture buttons.



Font

The Font selection allows you to choose a font for the button's text. You should select a font that is installed on your portable terminal.

Attributes

The Attributes section allows you to include Bold, Italic, Underline, or Strikeout attributes to your font choices. A check next to the attribute turns that attribute on.

Height & Width

The Height and Width font settings

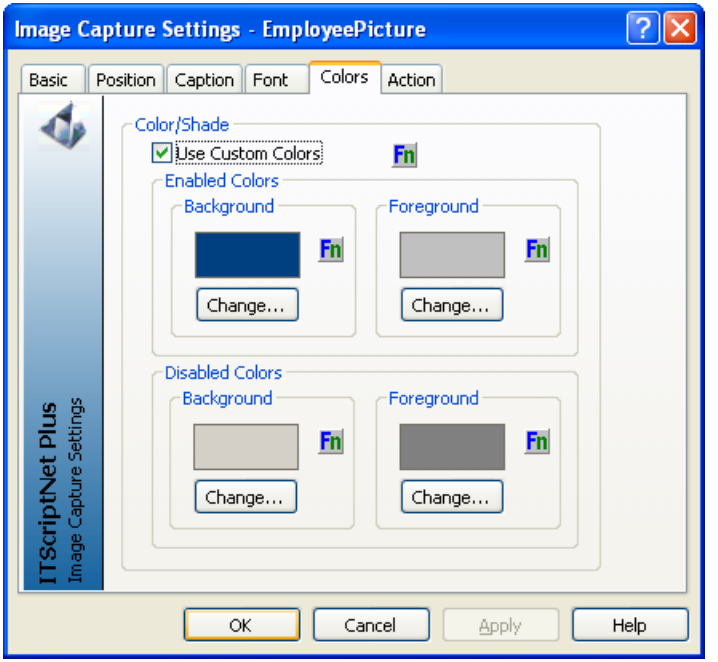
specify the Height and Width of the font size in pixels. The Font Preview section will reflect the Height and Width specified. If you use a Font Width of zero, Windows will select an appropriate default width.

Style

The Style setting allows you to apply a pre-defined Font and Color set to this element. The Style must have been created with the Style Editor on the Program menu. Any changes made to the Style in the Style Editor will automatically be applied to this element. If you change any Font or Color setting on the element, the Style will be removed.

Colors Tab

The Colors tab on the Image Capture Settings screen provides an opportunity for further customization of the image capture button. The colors will only be effective for portable terminals that have a color display. If the 'Use Custom Colors' option on the Colors tab is checked, the colors specified on the tab will be used.



Enabled Colors

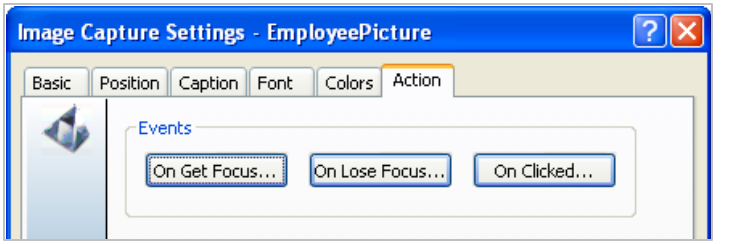
The Background color and the Foreground (Text) color on the image capture button can be specified independently. The enabled colors are used when the image capture button is enabled. Click on the Change… buttons to bring up the Color screen to specify the enabled colors.

Disabled Colors

The disabled colors are used when the Image Capture button is disabled and not able to be clicked. Depending on the terminal, the disabled colors may be overridden by the operating system of the terminal and the disabled colors may not be displayed as designed.

Action Tab

Events



The Image Capture element has several special event-driven scripts that allow for customized behavior.

- The **OnGetFocus** script runs when the element receives the focus. This can happen if the user clicks on the element, tabs to the element, if the focus is set to the element in a script, or can occur when a prompt is displayed if the element is the first element on the prompt.
- The **OnLoseFocus** Script runs when the element loses the focus, or in other words, when the focus is moved to another element.
- The **OnClick** Script runs whenever the element is tapped or clicked.

Digital Ink Element

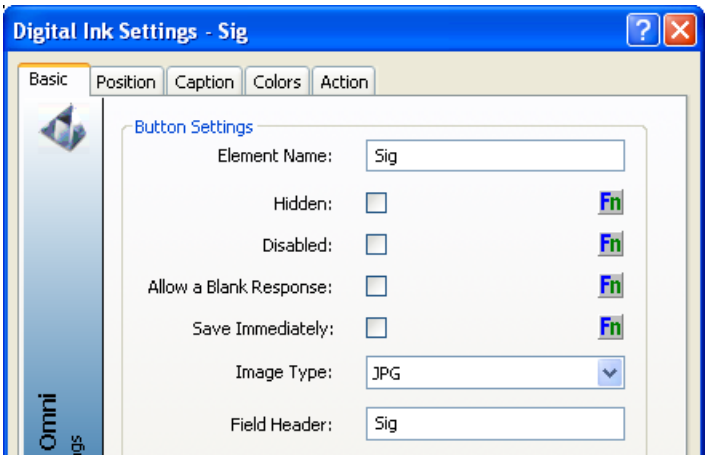
The Digital Ink element allows you to collect signatures or other drawn information from the terminal's screen. The digital ink element has many characteristics in common with the image capture element in that both types of elements produce images for the data collected.

To add a Digital Ink element to a prompt, Click on the Digital Ink element 

from the element list on the right side of the main Generator Application screen. You can turn on the elements list by checking the Elements menu under the View main menu item. After you click the Digital Ink element, the Digital Ink Settings screen will be displayed. You will need to specify the settings for the Digital Ink element. The key settings are the Element Name and the image file type. These key settings and all the Digital Ink element settings are discussed below.

Basic Tab

The Basic tab on the Digital Ink Settings screen is shown here.



Element Name

Digital Ink elements, like all elements, must have a name. When you first create the element the name will be displayed as "DigitalInk1". You may change the name to a valid element name. Element names can be up to 20 characters in length and must be unique within the prompt. The following characters may not be

used on prompt names: # $ @ _ - * . / = < > ( ) & as they are reserved.

Hidden, Disabled

The Hidden setting will cause the Digital Ink element to be hidden when checked. The Disabled setting will cause the element to be disabled. A Digital Ink element that is hidden and/or disabled cannot be clicked and will not be able to capture an image.

Allow a Blank Response

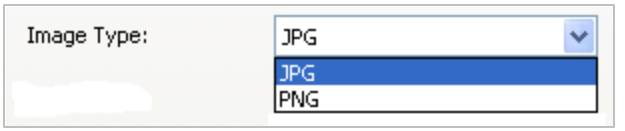
The response for a Digital Ink element is the image that was captured. The Allow a Blank Response setting determines in the Digital Ink element is allowed to have a blank image (element skipped) as its response.

Save Immediately

This field controls whether the actual Image File collected for this Digital Ink element is created when the prompt is Accepted (Immediately), or when the Collected Data Record is written to the file. By default, Digital Ink is not saved Immediately.

Image Type

The Image Type setting controls the format of the image that is captured. The two choices are JPG or PNG.



Database Field/Column Header/Field Header

This setting specifies where the data collected for this element is stored after it is sent to the PC and processed. Depending on the settings specified in the *Configure Receive* screen, this field may have different meanings.

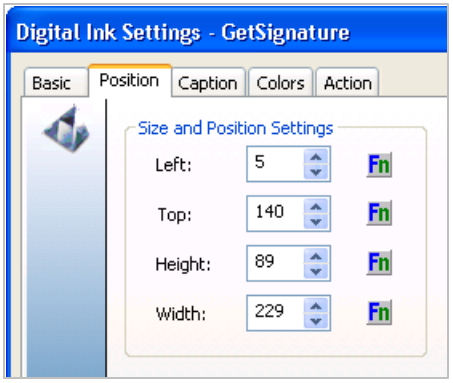
If the program is configured to send its data to a Microsoft Access file or an ODBC data source, the target database field must be set up properly so the data collected can be sent to the desired field. A field from the target database must be selected from the list of fields. The fields in the list are the fields defined for the database and table that are established on the *Configure Receive* screen. The <Ignore> selection can be chosen and the collected data for the Digital Ink element will be ignored when the collected data is sent to the database. Fields are output to Access and ODBC if the field specified here exists. The name of the image will be stored if the field selected is defined as a text field in the database table. If the field selected is an OLE object, then the actual image collected will be embedded in the database table.

If the program is configured to send its data to a Microsoft Excel file, this setting will become the column header for the data. For Excel files, if this field is left blank, the element

name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen. The file name of the image will be stored in the Excel file.

If the program is configured to send its data to a text file with a header, this setting will become the field header for the data in the output file. For text files with headers, if this field is left blank, the element name will be used as the column header. To skip this field and not output it, you will need to use the *Customize Field Layout* screen accessible from the *Configure Receive* screen. If the program is configured to send its data to a text file with no header information, then there will be no Field Header setting on the Digital Ink Settings screen. The Digital Ink element will store the file name of the captured image as its data if the *Configure Receive* screen is set up for a text file.

Position Tab



The Position tab on the Digital Ink Settings screen controls the size and position of the Digital Ink element on the prompt. You can also control the size and position of the element from the main design window. You can move the Digital Ink element by selecting it and holding the mouse while you move the element to the desired location. You can also resize the element by selecting it and dragging the resize rectangles.

Left Position

The Left Position setting specifies the horizontal location in pixels of the upper-left corner of the Digital Ink element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Left Position setting is set to 0, the Digital Ink element will be all the way to the left of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the left position to a value larger than the screen, the element will be located off the prompt screen.

Top Position

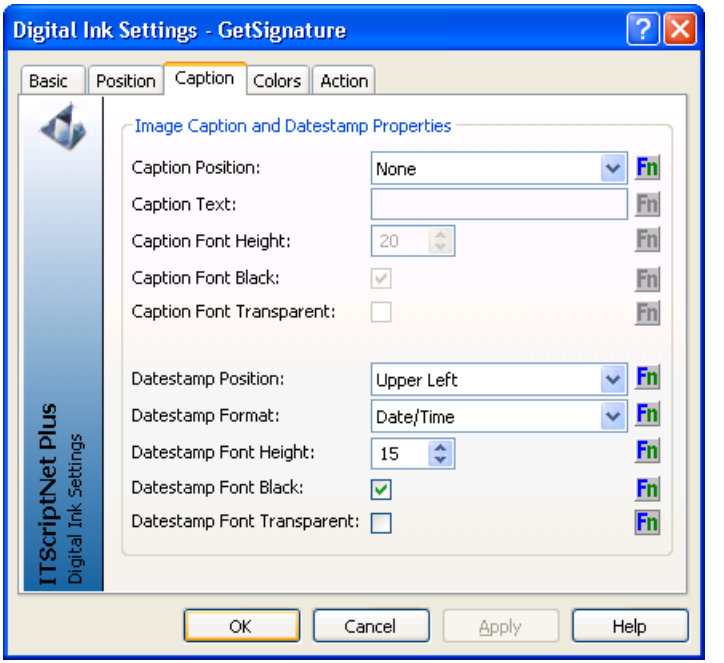
The Top Position setting specifies the vertical location in pixels of the upper-left corner of the Digital Ink element. The top left-hand corner of the main design screen is located at coordinates 0,0. Thus, if the Top Position setting is set to 0, the Digital Ink element will be all the way to the top of the prompt. The limits of the prompt screen depend on the type of portable terminal. There is also no restriction that requires the element to be located on the prompt screen. If you set the top position to a value larger than the screen the element will be located off the prompt screen.

Height and Width

The Height setting specifies the height in pixels of the Digital Ink element. The Width setting specifies the width in pixels of the Digital Ink element.

Caption Tab

The Caption tab allows you to customize captions that can optionally be superimposed on the digital ink image. There are 2 possible pieces of information to include. One piece of information is a general caption that you can specify. The other is a date/time stamp. You can choose to use neither, one, or both pieces of information to add detail to your digital ink images.



Caption Position

The Caption Position allows you to choose where in the image to display the caption text. If you choose "None", the caption will not be displayed on the image. The other choices are Upper Left, Upper Center, Upper Right, Lower Left, Lower Center, and Lower Right.

Caption Text

The Caption Text is the text that you want to be superimposed on the digital signature image you capture. You can use a script to create the text dynamically at run-time.

Caption Font

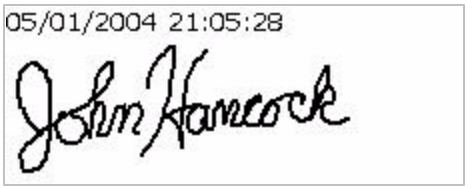
There are 3 settings related to the caption's font. The Caption Font Height is the number of pixels high the caption text will be. The width, then, is calculated automatically. The Caption Font Black setting will make the caption text black if checked or white if not checked. The Caption Font Transparent setting will make the rectangle enclosing the caption text transparent if checked. If the Caption Font Transparent setting is not checked, the caption text will appear on a solid white or black rectangle, the opposite of the font color.

Datestamp Position

The Datestamp Position works the same as the Caption Position. You can choose not to include a Datestamp or you can choose the location on the image for the Datestamp to be written.

Datestamp Format

This setting allows you to choose the format for your date/time stamp. You can choose to include both the date and time, just the date, or just the time. The image shown at the right is an example of an image that was captured from a Digital Ink element with a Date/Time Stamp in the Upper Left corner. The Font is black on a white background with no transparent font area.

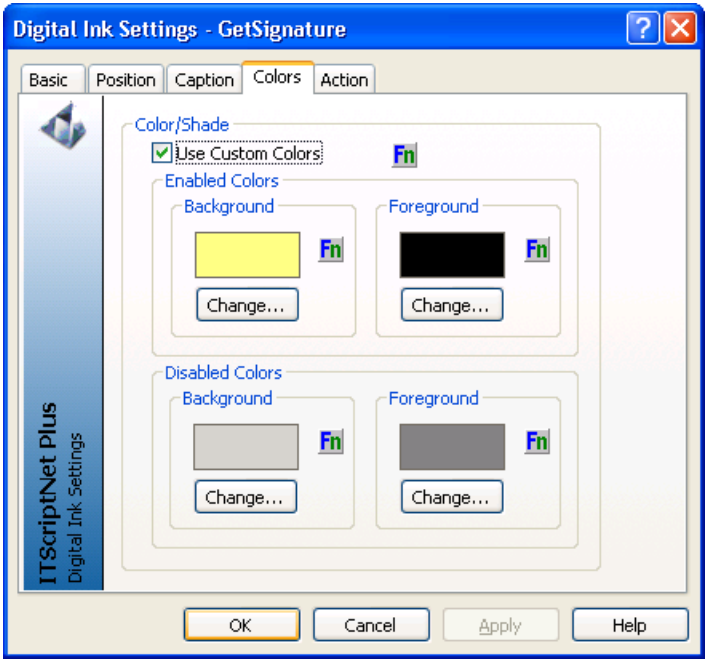


Datestamp Font

There are 3 settings related to the datestamp's font. The Datestamp Font Height is the number of pixels high the datestamp text will be. The width is calculated automatically. The Datestamp Font Black setting will make the datestamp text black if checked or white if not checked. The Datestamp Font Transparent setting will make the rectangle enclosing the datestamp text transparent if checked. If the Datestamp Font Transparent setting is not checked, the datestamp text will appear on a solid white or black rectangle, the opposite of the font color.

Colors Tab

The Colors Tab on the Digital Ink Settings screen provides an opportunity for further customization of the Digital Ink element. The colors will only be effective for portable terminals that have a color display. If the 'Use Custom Colors' option on the Colors tab is checked, the colors specified on the tab will be used.



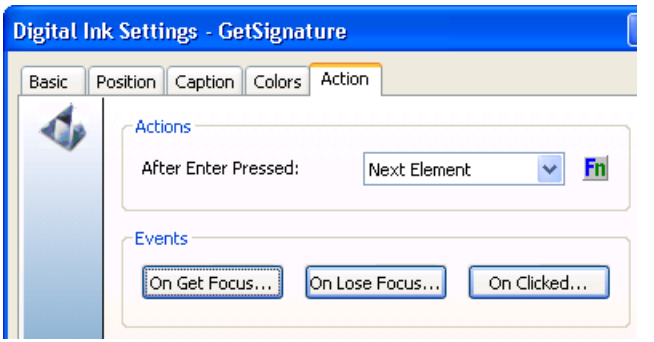
Enabled Colors

The Background color and the Foreground (Text) color on the Digital Ink element can be specified independently. The enabled colors are used when the Digital Ink element is enabled. Click on the Change… button to bring up the Color screen to specify the color. You can select from the basic colors on the left, select a color from the color matrix on the right, specify Hue/Saturation/Luminosity, or specify Red, Blue, and Green values to specify a custom color.

Disabled Colors

The disabled colors are used when the Digital Ink element is disabled and not able to be clicked. Note that Windows CE or PocketPC may override the disabled foreground color.

Action Tab



The Action tab contains the setting to control the behavior of the Digital Ink element after a response has been registered and if the user presses the Enter button. This tab also contains the access to the in-prompt scripts that can be used to customize behavior when the Digital Ink element Gets the Focus, Loses the Focus, or is clicked.

After Enter Pressed

There are 3 choices as to how the element will behave after the Enter button is pressed. The default behavior is for the element to do nothing. You can also choose to have the element advance the focus to the next element on the prompt. The order of the elements is determined on the prompt's settings screen. The last choice is to have the Enter key act as the acceptance for the entire prompt and thus the program advances to the next prompt as defined on the prompt's settings screen.

Events

The Digital Ink element has several special event-driven scripts that allow for customized behavior.

- The **OnGetFocus** script runs when the element receives the focus. This can happen if the user clicks on the element, tabs to the element, if the focus is set to the element in a script, or can occur when a prompt is displayed if the element is the first element on the prompt.
- The **OnLoseFocus** Script runs when the element loses the focus, or in other words, when the focus is moved to another element.
- The **OnClick** Script runs whenever the element is tapped or clicked.

Image and Text Capture in Single-Prompt

If your terminal is equipped with an **IT*Script*Net**–supported imager, you can capture images and text as part of the data you collect. Select Image Capture or Text Capture as the prompt's Input Source on the *Prompt Settings* screen.

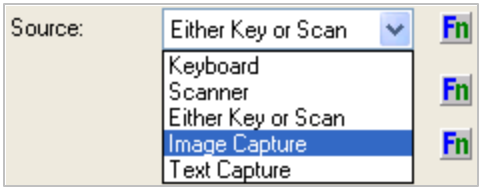


Image Capture

The terminal will take a 256-grayscale JPG image using the imager.

Text Capture

The terminal will take a 2-color (black and white) PNG image using the imager. This image is suitable for signature captures and other text capture. The image is optimized for clearly readable text. Text Capture is available only on the WinCE and PocketPC terminals.

Image Download

When the collected data is received by the PC from the terminal, the images are handled in one of two ways.

- **Image file download**
  If you select Text File or Excel format, or if the field in an Access or ODBC database is Text format, the image files will be placed in the same directory as the processed data file, and the field in the data file will contain the file name of the image.
- **Image embedding**
  This is an advanced feature. If you are collecting to Access or ODBC and the field is of type OLE Object or IMAGE (ADO adLongVarBinary), the image will be embedded in the database and the image file (originally stored in the directory with the .itb file) will be deleted. Note that the image is embedded as a BLOB, not as an OLE object. You will not be able to place the image on an Access report, but you will be able to display it using software that recognizes the image, such as Crystal Reports.

Image filenames

The filename for the image is constructed using the terminal alias and a datetime string. This reduces the chances of duplicate image filenames from different terminals.

Image and Text Capture in Multi-Prompt

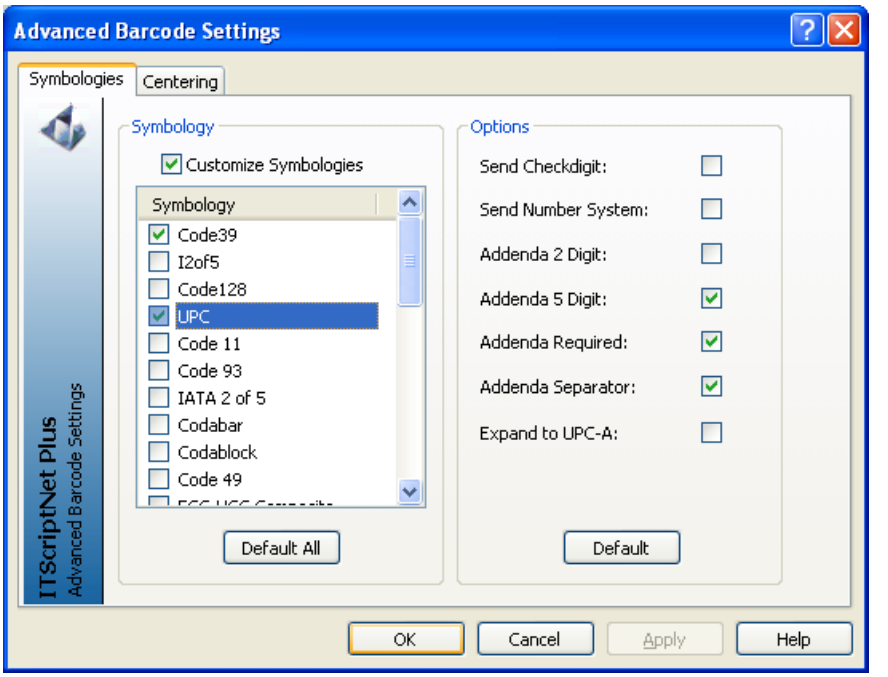
If your terminal is equipped with an imager, you can capture images and text as part of the data. To capture images with Multi-Prompts, simply include an Image Capture element on your prompt. Please refer to the section in this User Guide for Image Capture elements for more details.

Advanced Barcode Settings

This section applies to specific terminal clients that are supported by **IT*Script*Net** that also support the following features. If your terminal is not supported these features will be disabled. The Advanced Barcode Settings screen allows you to fully customize bar code scanning as a response to the data input element. The screen is accessed from the button next to the Symbology Settings on any element that is able to accept scans as a response type.

Symbologies Tab

The portable data collection terminals can be configured with scan engines with different capabilities. Each engine supports a different set of symbologies, and various options within those symbologies. Using the basic Symbology selection on the *Prompt Settings* screen allows you to specify a symbology that must be scanned, but all scans will use the scan engine defaults. The *Advanced Symbology Settings* screen provides access to the underlying scan engine settings, and gives greater control of exactly how barcodes are to be scanned, validated, and how data will be returned.



The *Advanced Symbology Settings* screen is accessed by clicking the button with the barcode icon to the left of the Symbology drop-down box on the *Prompt Settings* screen. Please note that this screen is only available when the **Any/Multiple** symbology selection is chosen. If a specific symbology is selected, this button is disabled.

Customize Symbologies

This check box controls whether the customized symbology settings will be effective. If this box is not checked, no customization will be performed, and all barcodes will be enabled and scanned according to their defaults. If checked, the scan engine will be configured according to the settings specified on the rest of the Advanced Symbology Settings screen.

Caution:
Some Portable Terminals take a significant amount of time to configure their scan engines. If Customized Barcode Symbologies are enabled, you may see a performance impact on the terminal. Specifically, terminals with slower processors can sometimes take several seconds to configure the scan engine for customized options.

Symbology

The symbologies supported by the terminal's scan engine are listed here. Each symbology can be enabled or disabled by checking or clearing the checkbox to the left of the symbology in the list. This allows you to specify a single symbology or multiple symbologies as valid symbologies for the prompt.

When you click on a symbology name, the options available for the symbology appear to the right in the Options area.

Options

The Options area displays the scan engine options for the selected symbology on the selected terminal type. Each terminal has a different set of options. For more information on specific symbologies and options, please consult your data collection terminal documentation.

Options can be either Checkboxes or Input fields. If an option is a checkbox, you can check it to enable the option, or uncheck it to disable the option. Examples of checkboxes may include enabling check digits, whether to return start and stop characters in the data, etc.

If an option is an input field, you can type a numeric value for the field. Examples of numeric values may include maximum or minimum lengths for a symbol.

Default

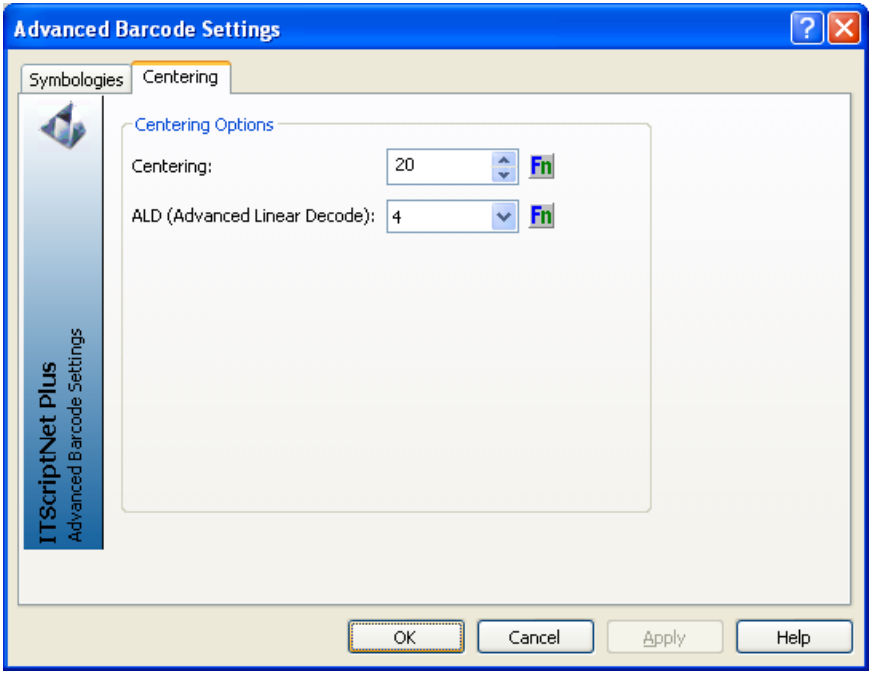
Click the **Default** button to reset the values of the Options to the scan engine defaults for the selected symbology.

Default All

Click the **Default All** button to reset the values of the Options to the scan engine defaults for ***all*** symbologies. This action does not change the status of the checkboxes for which symbologies are enabled. All symbologies will reset to the default parameter values regardless of whether the symbology is checked or not.

Centering Tab

This tab covers both Centering and Advanced Linear Decoding options for the mobile computers equipped with HandHeld Products Imagers.



Centering

The centering setting will turn on the centering feature of HandHeld Products imagers. The centering feature defines a region relative to the center of the image. The region is defined in pixels and is X pixels wide and X pixels high around the center of the image where X is the centering value. If a symbology to decode is found to be within or touching that region, the symbol will be decoded. If the symbol is not found in or touching this region, the symbol will be ignored. A centering setting of zero will disable this feature and any symbol within the imaged area will be decoded. The centering feature is useful when you have an application where multiple barcodes may be in view of the imager at the same time and you want to limit the barcode that is decoded to be the one in the center of the image. A value of 20 for the centering option works well for these applications.

ALD (Advanced Linear Decode)

The ALD setting turns on the HandHeld Products Imager's Advanced Linear Decode feature. The ALD provides faster decode performance on linear (1-D) barcodes. A value of 0 turns ALD off. The Values from 1-6 will determine the vertical range around the center of the imager that ALD uses to locate the symbol to decode. A value of 1 is the smallest region and a value of 6 specifies the full vertical height.

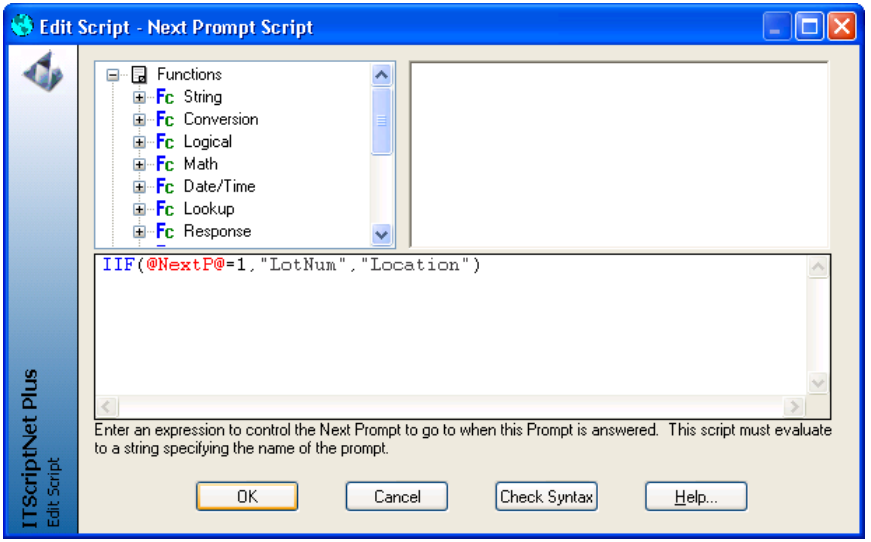
In-Prompt Scripts

IT*Script*Net's Prompt Settings give the designer of a data collection program a great deal of flexibility to customize a data collection solution with no programming. In **IT*Script*Net Batch Plus** and **OMNI** editions, the variety of element types and the broad range of element settings available add another dimension of flexibility in program design, again with no programming. With **IT*Script* Net**'s In-Prompt Scripts, the possibilities are virtually endless.

In-Prompt Scripting adds an amazing level of power to **IT*Script*Net** programs. Using these simple-to-write scripts, you can control any setting in the program at run-time. This allows the data collection program to adapt to the inputs made on the terminal and to change its behavior accordingly.

Setting Scripts

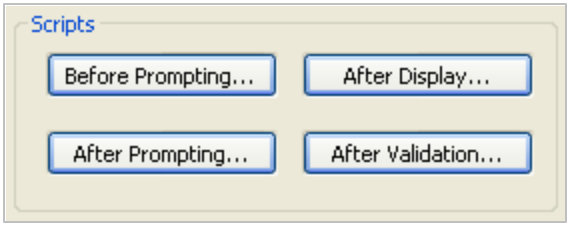
Almost every prompt or element setting in **IT*Script*Net** has an in-prompt script that allows the value for the setting to be determined programmatically. This means that the setting value can be based on data that has been collected or the value of program variables as opposed to being a value set at design-time.



The In-Prompt script for a setting script can contain any number of lines of code. However, the last line of code in the script must evaluate to a value that is appropriate for the setting. The last line of the script evaluates during run-time to a value used to set the property value. For example, if the In-Prompt Script is a script to change the Max Length of the prompt, then the script must evaluate to a number appropriate for a Max Length. If the script is for the barcode symbology, then the In-Prompt Script must evaluate to one of the symbology constants. The bottom of the script editor screen provides information as to the type of expression that is required for the setting. In the example shown, the In-Prompt Script is for the Next Prompt setting. Thus, the result of the script expression must evaluate to a name of a prompt (string).

Prompt-Level Scripts

The prompt-level scripts allow you to add additional functionality to your programs by allowing scripts to be run at specified points in the sequence of prompt evaluation.



Before Prompting

This script is executed before the prompt is displayed to the terminal operator. You can use this script to generate the text to be displayed for the prompt, or to determine values to use for elements for Multi-Prompts.

After Display

The After Display script runs immediately after the prompt is displayed to the user. For Multi-Prompts, it is a good place to add functions that control the element that has the focus.

After Prompting

This script is executed after the operator collects the data for the prompt but before the built-in validations are checked (i.e. Mask, Range Checking, etc). You can use this script to modify the collected data or to perform additional custom validations.

After Validation

This script is executed after the built-in validations are checked. You can use this script to modify the collected data or to perform additional custom validations.

Event Scripts

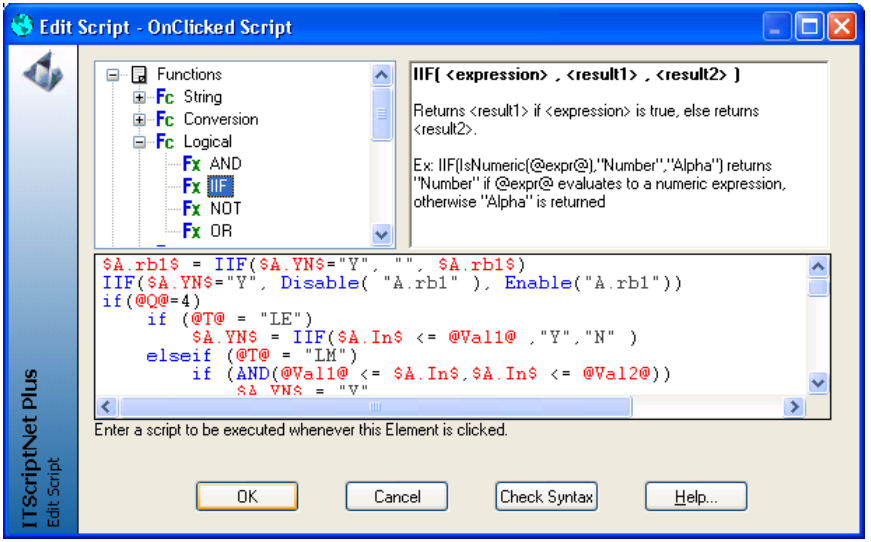
Event scripts are another type of in-prompt scripts available for Multi-Prompt elements. These scripts are attached to events for elements. Most of the element types have event scripts available when the element receives the input focus, loses the focus, or is clicked by the user.

Script Editor Screen

Clicking on an In-Prompt Script button will bring up the *Edit Script* screen where the In-Prompt Script for a specific prompt property or advanced property can be edited. The Script Editor screen is resizable, and saves its size and position if they are changed.

Script Element Tree

The top-left area contains a dynamic list of the functions, lookups, responses, etc. available to the script. Double-clicking an element in the list will insert the item into the script. This always-available reference makes writing the In-Prompt script easy since you do not have to memorize functions and programming syntax.



Function Definition Area

The top-right area displays the name, arguments, and description for the function selected on the left. An example using the function is also provided.

Script Area

The bottom portion of the Script Editor Screen is the area to write the in-prompt script. The script can be any number of lines long and can include functions, variables, keywords, etc.

Note: If the script is used to determine the value of a prompt or element setting, then the last line of the script must evaluate to a value that is valid for that setting. If the script is a prompt-level script or an event script, there is no restriction on the last line of the script.

Syntax-Coloring

The In-Prompt script is syntax-colored to assist in writing the scripts. Key words and function names are in blue. Variable names, including responses to prompts and user-defined variables, are in red. Strings are gray and constants are green. The syntax-coloring helps to identify the elements in an in-prompt script more readily and makes the script easier to read and use.

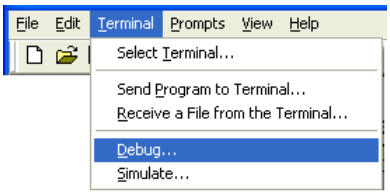
Script Comments

You can insert comments in your scripts by starting the comment line with a semi-colon. The entire line in the script will be disregarded when the script runs. You cannot start a comment in the middle of a line in the script.

Check Syntax

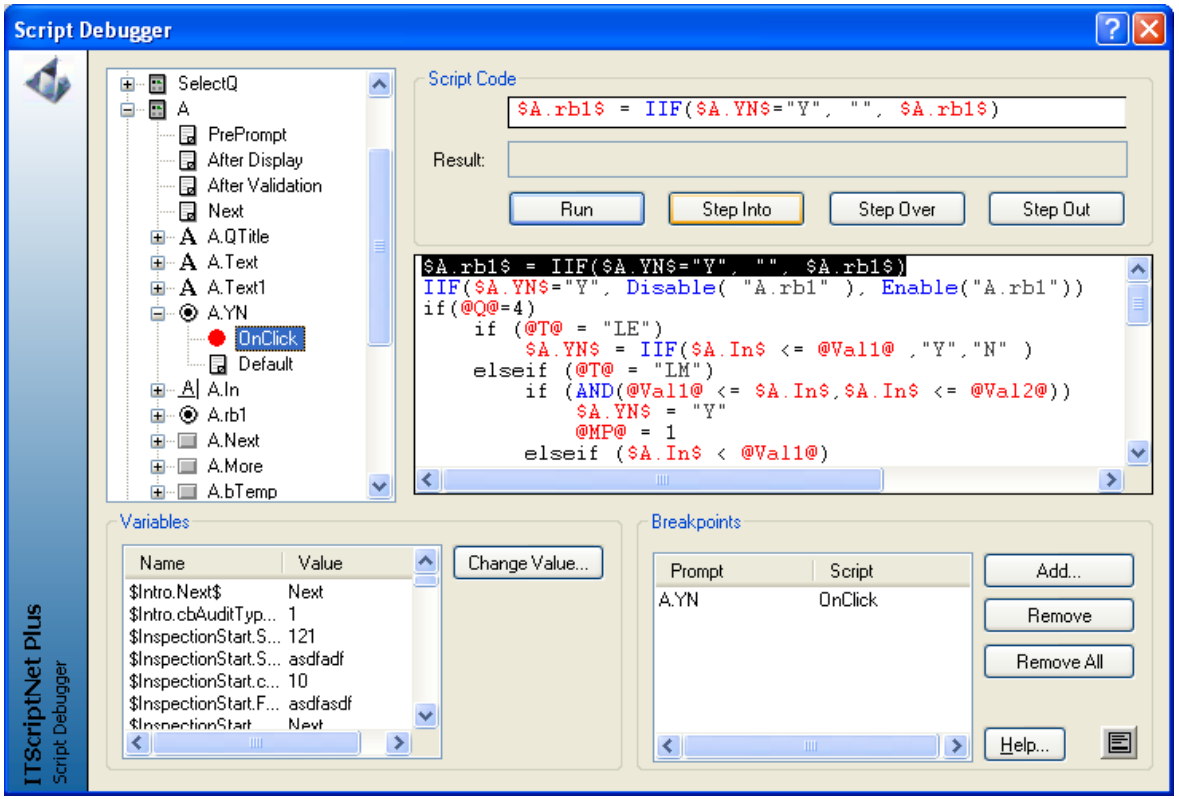
The **Check Syntax** button will cause **IT*Script*Net** to run a trial evaluation of the In-Prompt Script and will either inform the user that the expression evaluated OK or will inform the user of a syntax problem. The **Check Syntax** button only checks syntax and is not able to truly and fully test the expression since there are no real values for the script elements at design time.

Debugging In-Prompt Scripts

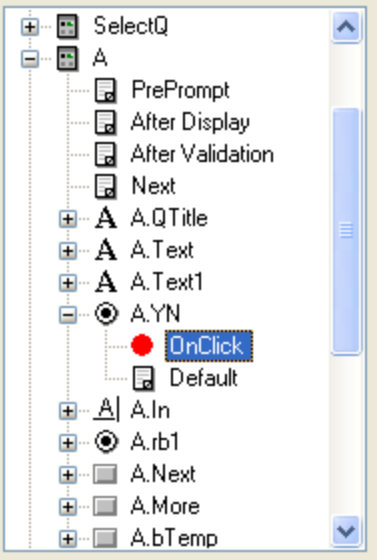


IT*Script*Net has a script debugger that can be used to assist in the development of your in-prompt scripts. The Debugger works in conjunction with the simulator. The Script Debugger screen allows you to stop scripts while they are executing so that you can step through them and investigate in detail what variables or actions are occurring so that you may find any problems you might have with your script. To start the debugger, select *Debug* from the *Terminal* menu.

The *Script Debugger* screen is shown below.

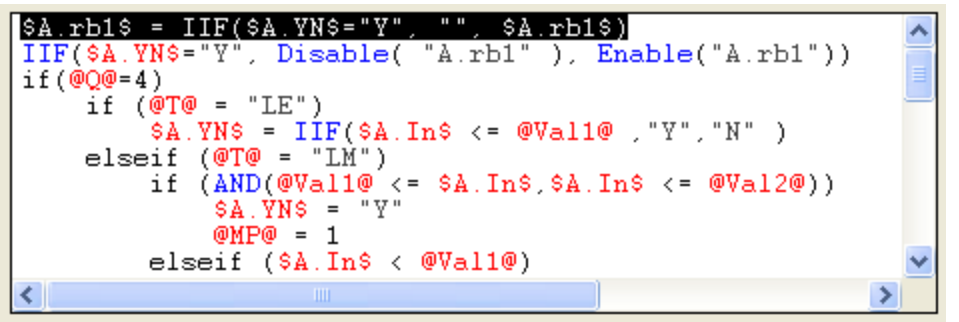


Prompt, Element and Script Tree



On the left side of the screen there is a list of all prompts and elements in the program and the in-prompt scripts that have been created. You can click the [+] sign next to a prompt name to expand the list to see the In-Prompt scripts that have been defined within the prompt and the elements for that prompt. You can further click on an element to see its In-Prompt scripts. In the example here, the prompt named "A" has prompt-level scripts and several elements. The radio button named "YN" is shown expanded to reveal that it has two In-Prompt scripts—the OnClick script and the Default value script. If a breakpoint has been set for a script, the icon for the script will change to a small red circle. When a breakpoint is reached, the icon for the script will be a blue arrow. The OnClick script in the example has a red dot to show that a breakpoint has been set for this In-Prompt script.

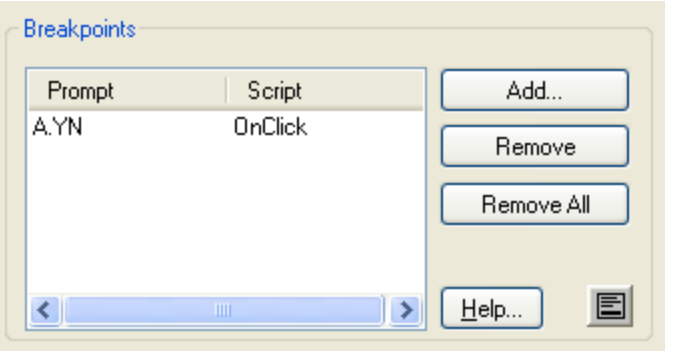
Code Area



```
$A.rb1$ = IIF($A.YN$="Y", "", $A.rb1$)
IIF($A.YN$="Y", Disable( "A.rb1" ), Enable("A.rb1"))
if(@Q@=4)
     if (@T@ = "LE")
          $A.YN$ = IIF($A.In$ <= @Val1@ ,"Y","N" )
     elseif (@T@ = "LM")
          if (AND(@Val1@ <= $A.In$,$A.In$ <= @Val2@))
               $A.YN$ = "Y"
               @MP@ = 1
          elseif ($A.In$ < @Val1@)
```

When you click on a script name in the list, you will see the actual script code in the window in the center of the *Script Debugger* screen. This code listing uses the same syntax coloring as the Script Editor. The script cannot be edited in the Debugger, however.

Breakpoints



The Breakpoints area allows you to set a breakpoint on any script. When the program is running in the simulator, execution will stop when a script with a breakpoint is reached. This gives you a chance to examine variables, step into statements, etc. The Breakpoints

section of the debugger screen always displays the current list of breakpoints. Note that you cannot set a breakpoint on a specific statement within a script, just at the start of the script itself.

Add Breakpoint

To add a breakpoint, select an In-Prompt Script from the tree of prompts, elements, and scripts in the upper left portion of the debugger screen. Click on the **Add** button in the Breakpoints section to add the selected script to the list of breakpoints. The script icon in the tree will change to a red circle to indicate that a breakpoint has been set.

Remove Breakpoint

The **Remove** button removes the currently selected breakpoint from the breakpoint list. The icon for the script will change back to normal.

Remove All Breakpoints

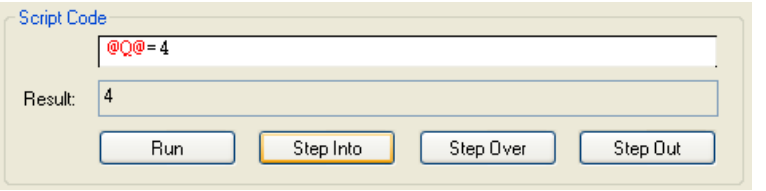
The **Remove All** button removes all breakpoints.

Simulator button

Click the **Simulator** button to bring the simulator window to the front. This button is only valid once the program execution has started.

Script Code

The Script Code area on the upper right displays the current line of script code when stepping through a script. The script line is displayed on the top line, and the result that will be returned is displayed on the bottom line.



You can control the execution of the script using the **Run**, **Step Into**, **Step Over,** and **Step Out** buttons. These buttons are active when the script execution has stopped at a breakpoint.

Run

Click the **Run** button to start the program in the debugger or to continue execution. The program will run until the next breakpoint.

Step Into

Click the **Step Into** button to execute a portion of the current script. This button causes the debugger to step into nested functions.

Step Over

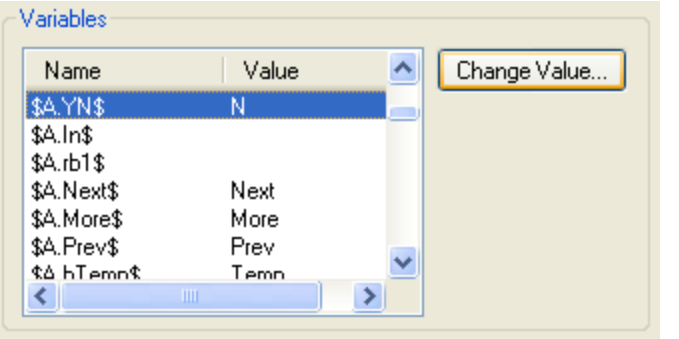
Click the **Step Over** button to execute the current statement in its entirety and stop on the next line in the script. Note: If you step over the last line of a script, the program will continue to run until the next script is encountered.

Step Out

Click the **Step Out** button to execute the current script in its entirety and stop on the next script.

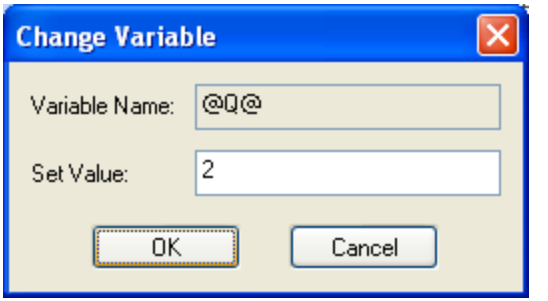
Variables

The Variables area displays the list of all variables currently defined and their values.



Change Value

You can change a value of a variable by selecting the variable from the variable list and then clicking on the **Change Value** button. This will bring up the *Change Variable* screen.

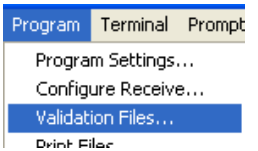


Enter the new value for the prompt and click the **OK** button to save the new value for the variable and return to the *Script Debugger* screen. Click the **Cancel** button to keep the value as it is and return to the *Script Debugger* screen.

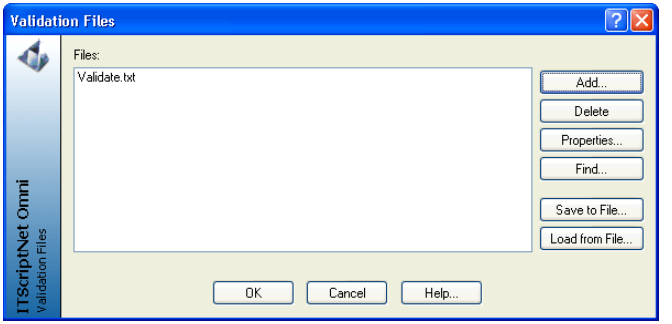
Configuring Validation Files

An **IT*Script*Net** data collection program can use multiple validation files. Validation files are used so that responses to prompts can be validated against a set of data to determine if the response is acceptable. Use of validation files helps assure that the data collected is accurate. Each validation file to be used to validate data for a prompt within a program must be defined for the program on the *Validation Files* screen. You specify the file name and create fields within the file. It is recommended that the file name be 8 characters or less if the data collection program is to be run on a DOS-based terminal.

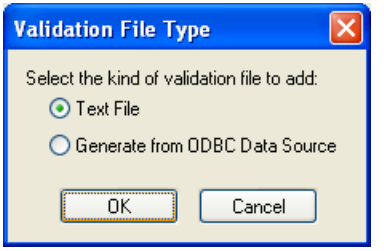
Validation File List



To add a validation file to the program, click on **Program** on the menu bar and select **Validation Files** form the drop down list. In the Validation Files window, click the **Add…** button. **IT*Script*Net** will ask what type of validation file to add: Text File or Generate from ODBC Data Source? Text file validation files are existing fixed-field length text files. These text files can be created by exporting data from Excel, Access, other data sources, or created in an application such as Notepad. If a validation file is no longer needed by the program, select the file from the list and click on the **Delete** button to delete the file from the list of validation files for the current data collection program.



The **Find…** button will allow you to browse for the selected validation file. This is useful if you receive a data collection program from another **IT*Script*Net** user who has a different directory structure and the validation file must be located in a different folder.



Long/Short Filename Note

When adding files to the Validation Files list, you will be warned if your filename is a long file name. DOS-based terminals only support short filenames. If you use long file names, they will be translated to short names before they are loaded into the terminal. Normally, this presents no problem, although there is a risk that the short file name may not be found on another PC. For this reason, we recommend that you use short file names whenever possible.

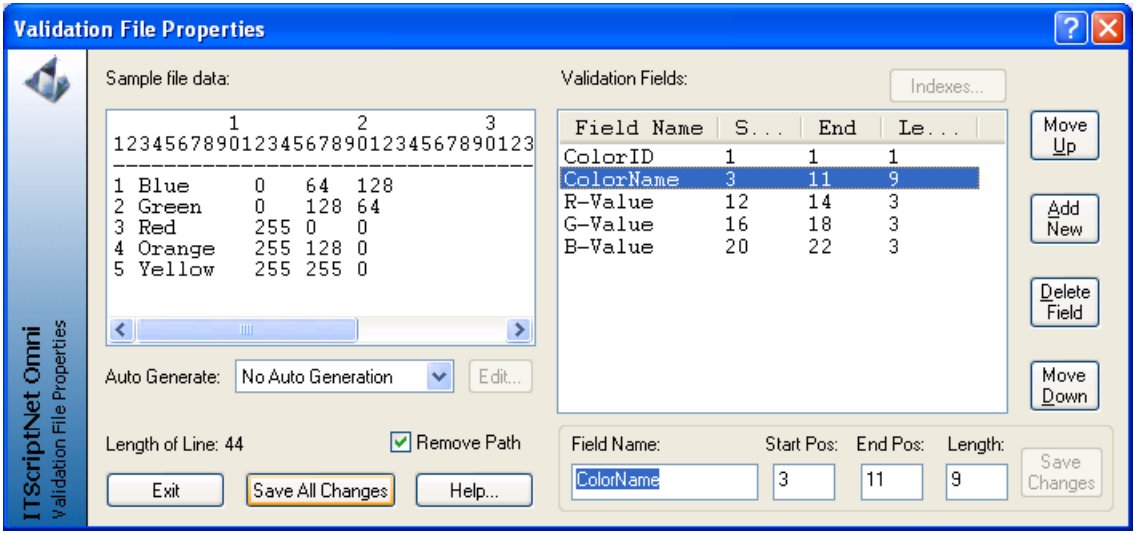
Save to File / Load From File.

If you want to use the same validation file structure in more than one program, you can save the file definition to an external file, and then load it into a different program. Selecting the **Save to File** button will allow you to select the filename to received the definition. Selecting **Load from File** will allow you to select an existing file and load the definition into the current program.

If the validation file name being imported matches the name of an existing file, you will be prompted that the name exists, and asked if you want to replace the existing definition with the one from the file, or cancel the load.

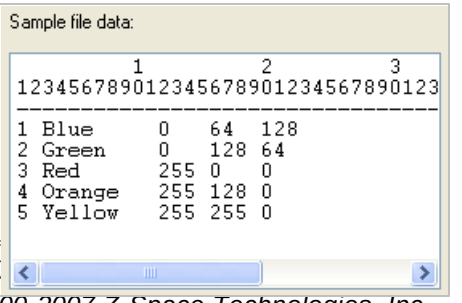
Validation File Properties

After adding a validation file to the list, the validation file must be defined by clicking on the **Properties…** button. The *Validation File Properties* screen will be displayed. This screen will allow you to define the validation file by specifying names and lengths for each field in the file. Save the Validation file settings by clicking on the **Save All Changes** button after making changes to the validation file properties.



Sample File Data

The left side of the screen shows a sample of your text file. The top lines displayed are a ruler to help identify the format of the text file.



Field Name

It is necessary to define fields contained within the text file. In the example shown, the first character in the text file is the identification number for the color. The name of the color is a separate field in the file. Each validation file field that you will need in your data collection program must be defined on the Validation File Properties screen so that the program can make the fields available for use to your program.

Start Position, Length

Along with the name of the field, the starting character position and the length of each field must be specified. In our example, the ColorName field starts at position 3 and ends at position 11. The ruler at the top of the sample data will help easily identify the starting and ending positions for each field.

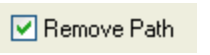
Add New, Delete Field

Click the **Add New** button to add a new field definition to the validation file. The area for specifying the name of the field and its start and end characters will default to be ready for your new field. After entering the information for the field name and the start and the end position or length, click the **Save Changes** button. The **Delete field** button will delete the selected validation field from the list.

Move Up, Move Down

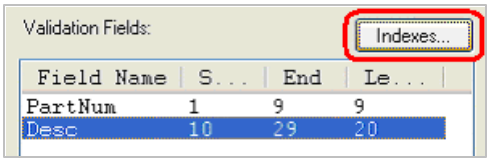
The **Move Up** and **Move Down** buttons adjust the order of the list of fields by moving the selected validation field in the list up or down.

Remove Path



Checking the Remove Path checkbox will cause the validation file to be stored without a path. The validation file will be assumed to be in the same directory as the ITB file. Use this option when you will be moving your ITB file and validation files to another PC with a different directory structure.

Validation File Indexes

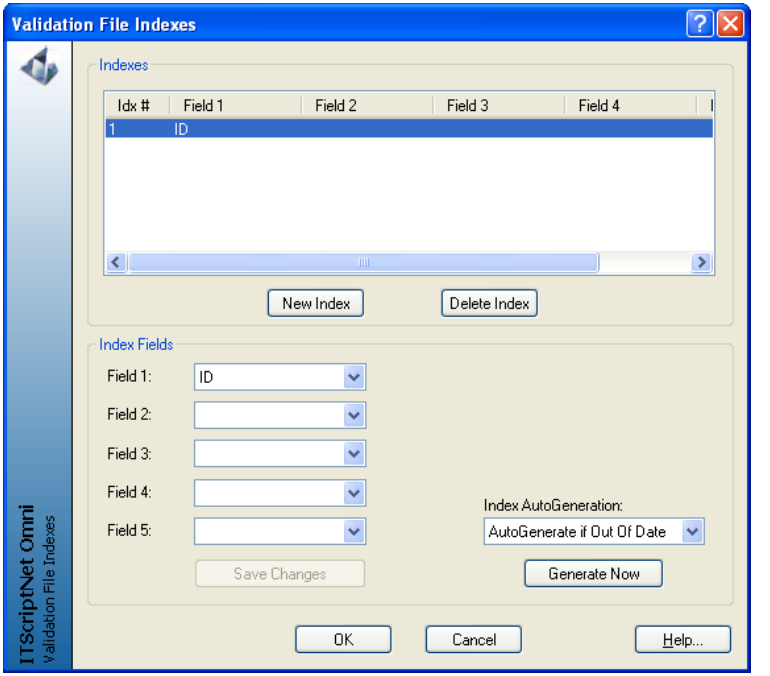


The *Validation File Properties* screen has an **Indexes** button. This button is disabled until you save your field changes. Click on the **Indexes** button to bring up the *Validation File Indexes* Screen. This screen allows you to define one or more index files for the validation file. Index files will improve the performance of lookups against large validation files. Indexed validation files that are large will realize a significant decrease in the time required for a lookup. Index files do take up file space on the terminal, so there may be some situations where index files cannot be

used. Very small validation files will benefit less from index files than larger validation files because the lookup speed is already short for small validation files.

New Index

To add an index to the validation file, click on the **New Index** button. This will activate the drop-down boxes in the Index Fields section of the screen. Select the field or fields for the index and click on the **Save Changes** button. Your new index file will be added to the list at the top of the screen.



Delete Index

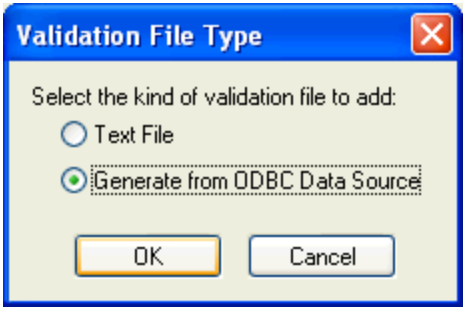
An index can be deleted from the validation file by selecting the index from the list at the top of the screen and clicking the **Delete Index** button.

Generate Indexes

Each index defined for a validation file will be generated to a separate file with an .ix? extension. You can generate an index manually by selecting the index from the list of indexes at the top of the screen and clicking on the Generate Now button. You can also have the index created automatically by selecting the appropriate *Index AutoGeneration* option. You can specify to generate the index file manually, on every upload, or only if the existing index file is older than the corresponding validation file. Index files are sent to the terminal with the validation file when a data collection program is sent to the terminal. It is recommended to allow the index files to be created automatically if out of date, so that the index file is always current and in-sync with the validation file.

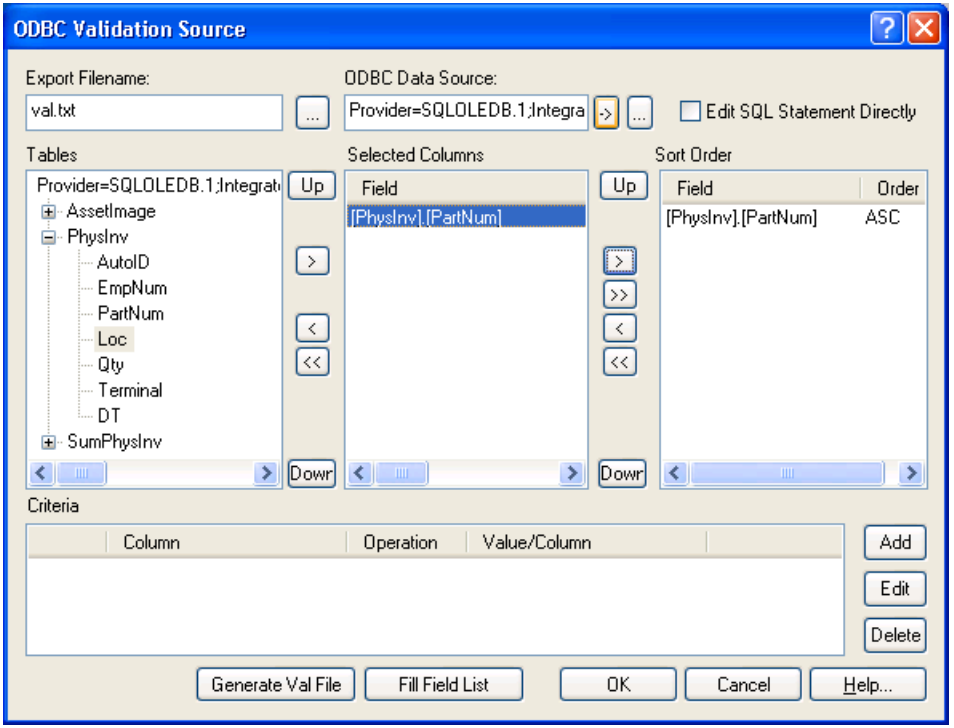
Auto-Generate Validation Files

In **IT*Script*Net**, validation files can be generated automatically from an ODBC data source. Automatically generating validation files can eliminate the need to create validation files for use by the data collection program you design in **IT*Script*Net**. From the *Validation Files* screen, clicking the **Add** button will bring up a screen asking whether you wish to add a validation text file or whether you want to add a validation file that is generated by **IT*Script*Net** from an ODBC data source. The Text File option will allow an existing text validation file to be added in the manner that was described in the preceding sections.

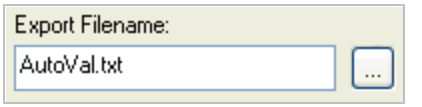


ODBC Validation Source Screen

Select the **ODBC** option and click **OK** to bring up the *ODBC Validation Source* screen as shown here.

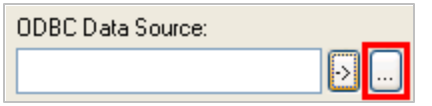


Export Filename



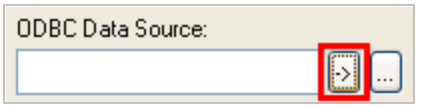
The Export Filename area on the screen is used to specify the name of the validation file that will be created (exported) from the ODBC source. The **'...'** (Browse) button can be used to browse for the file.

ODBC Data Source-DSN



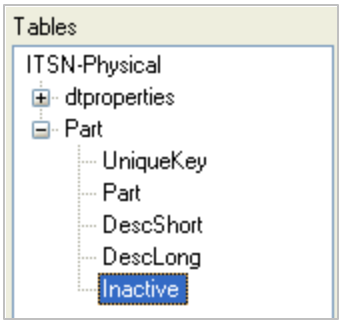
Select the ODBC Source for the validation file by clicking on the "..." (Browse) button and clicking on the data source in the list that is displayed. The ODBC Data Source must exist to be in the list. An ODBC Data Source is established using the ODBC Data Source Administrator utility that is installed with ADO (ADO is typically installed with **IT*Script*Net**, see the installation chapter for more information). The ODBC Data Source Administrator is accessible in Windows 95/98/Me from the Control Panel or from the Administrative Tools in Windows 2000 and XP.

ODBC Data Source-Data Link



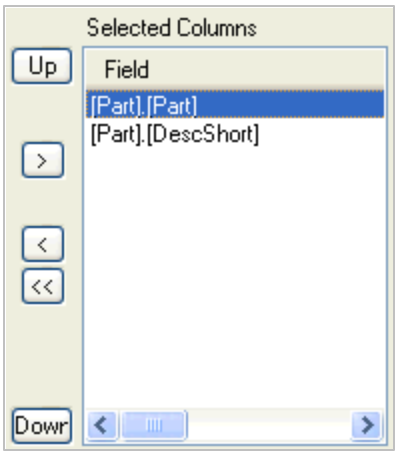
You can establish an ODBC connection without a DSN with the Data Link button. Click the Data link button and select the Provider and Connection settings. This will establish a direct connection string that **IT*Script*Net** can use to generate the validation file without a pre-defined DSN.

Tables



The Tables section of the screen is a list of tables contained within the ODBC data source. This explorer-style list will be filled with the table and field information from the ODBC data source once the data source is selected. Click on the + sign or double-click a table name to expand the list to see the fields in the table. Depending on the type of database for the ODBC connection, there may be system tables or other information included in this list.

Selected Columns

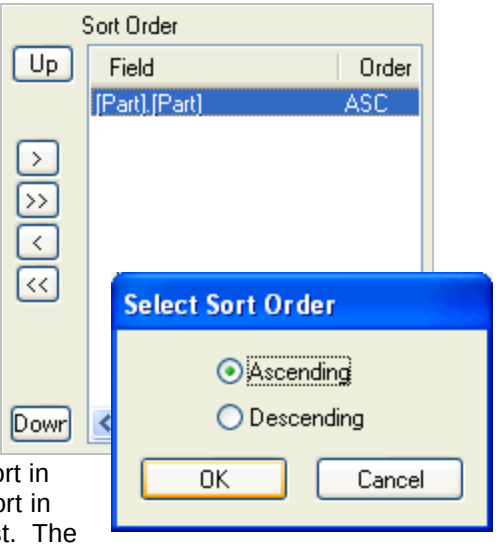


The selected columns list displays the columns, or fields, from the data source that will be included in the generated validation file. Double-clicking a field name in the Tables list will add the field to the Selected Columns list. You can also select a field and click the **'>'** button to add a field as a Selected Column. You can add all fields for a table to the Selected Columns list by double-clicking the table in the Tables list. Once the Selected Columns list has been filled, the order of the fields in the validation file can be set by using the **Up** and **Down** buttons to the left of the Selected Columns. If you need to remove a field

from the Selected Columns list, click the **'<'** button to remove the item. You can remove all the items by clicking the **'<<'** button. In the example, only the Part and DescShort fields were chosen to be included in the validation file.

Sort Order

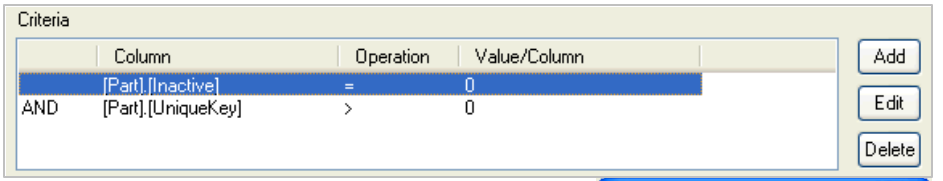
The Sort Order list on the right side of the screen displays the fields that will be used to sort the data. It is not required that there be fields in the sort order list, but a logical sort order will often be appropriate. In the example shown, the resulting validation file will be sorted by the Part field. To add a field as a sort order, double-click the field in the Selected Columns list or click the **'>'** or **'>>'** buttons to add the selected field or all fields to the sort list. You can change the sort order by using the **Up** and **Down** buttons to the left of the Sort Order list. You can remove items from the sort list by clicking the **'<'** button to remove the selected item or remove all items by clicking the **'<<'** button.



By default the fields selected as sort fields will be set to sort in ascending order. However, you can change the field to sort in descending order by double-clicking the item in the sort list. The *Select Sort Order* screen will allow an ascending sort order or a descending sort order to be chosen. Pick the order appropriate for the field and click OK to set a new sort order.

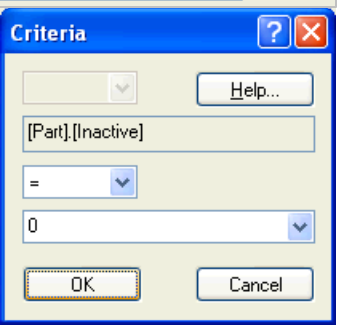
Criteria

The criteria area allows statements to be added to the criteria list that will limit the validation files to an appropriate subset of the data. In the example below, the criteria has been set so that only parts where the Inactive field is zero will be included in the validation file (only active parts will be included in the generated validation file).



Add Criteria

To add an item to the criteria list, select the field for the criteria from the Tables list and click the **Add** button. The *Criteria* screen will show the field selected for the criteria and will allow the statement for the criteria to be built. A comparison operator must be selected from the drop-down list in the middle of the screen and a value must be entered in the bottom field. The bottom field is also a drop-down list that can be used to select a field value to complete the criteria



statement. Click OK to close the screen and save the criteria to the list.

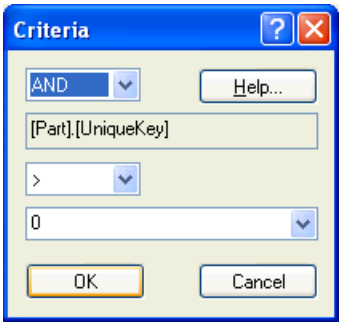
Edit Criteria

A statement in the criteria list can be edited by selecting the item and clicking on the **Edit** button. The *Criteria* screen will be displayed and the statement can be edited. Click **OK** to save the modifications and return to the *ODBC Validation Source* screen or click **Cancel** to abort the changes.

Delete Criteria

Select the criteria item in the list and click the **Delete** button to remove the selected item as a criterion.

Multiple Criteria



When adding the second (or higher) criteria to the criteria list, the **Add** button will display the *Criteria* screen. The difference is that the logical 'AND' or 'OR' must also be selected. If 'AND' is selected and there are two criteria, then both criteria must be true for the record to be included in the generated validation file. If 'OR' is chosen and there are two criteria then only one of the two criteria must be true for the record to be included in the generated validation file.

Be sure to add single-quotes around your data value if they are necessary. This is usually the case if you are matching a string or date type field.

Generate Val File

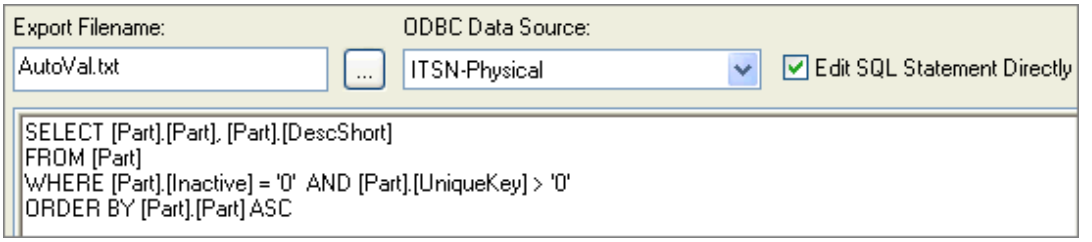
Click the **Generate Val File** button to cause **IT*Script*Net** to generate the validation file based on the Selected Columns, the Sort Order, and the criteria that have been selected or specified. Or, if the SQL Statement has been edited directly, the validation file will be generated based on the edited SQL statement. The validation file generated will be a text file named and located as specified in the Export Filename area in the upper left corner of the screen. The validation file can be used as a normal validation file and is ready for you to define its fields and indexes so you can use it in your prompts' Advanced Properties and In-Prompt scripts.

Fill Field List

The generation of the validation file is separate from the definition of the validation file that occurs on the *Validation File Properties* screen. The fields and field lengths must still be defined for the validation file even though the validation file is configured to auto-generate. The **Fill Field List** button pulls the generation of the validation file and the definition of the validation file together by automatically filling in the validation file field definitions according to the ODBC fields used to generate the file. This feature is a convenience to save the few minutes it would take to enter the validation field names and sizes on the *Validation File Properties* screen. If the fields for the validation file are already defined, the Auto-Generation process will use the field definitions already in place. If there are fields already defined for a validation file, the **Fill Field List** button will overwrite the previously defined fields.

Edit SQL Statement Directly

The *ODBC Validation Source* screen, with its several list areas, is a visual means of deriving a Structured Query Language [SQL] statement to use to obtain a set of records to include in the generated validation file. Check the checkbox in the upper right corner of the screen to



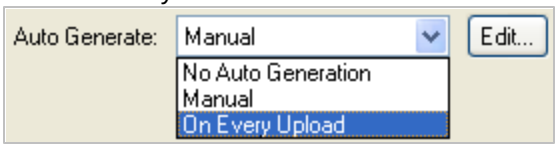
switch the screen into direct SQL edit mode. The example shows the SQL statement that results from the examples previously shown in this section. The "Select [Part].[Part], [Part].[DescShort] FROM [Part]" portion of the SQL statement reflects the table and fields chosen to be in the Selected Columns list. The "WHERE [Part].[Inactive] = '0' AND [Part].[UniqueKey] > '0' " portion of the SQL statement reflects the two criteria specified and the "ORDER BY [Part].[Part] ASC" represents the sort order. The main view of the *ODBC Validation Source* screen allows the SQL statement to be built visually, which is easier and faster than writing the SQL directly. Editing the SQL statement directly, however, allows highly complex SQL statements to be used to generate validation files. The SQL statements that are generated use Microsoft SQL syntax – the [tablename].[fieldname] syntax. Other databases, Oracle for example, use a different SQL syntax variant. The ability to edit the SQL directly allows adjustments to be made to the SQL statement to support any underlying database's SQL variations.

Regenerate SQL

The **Regenerate SQL** button will use the settings from the visual mode lists to reset the SQL statement. Click this button to undo any modifications that have been made to the SQL statement manually. If changes to the SQL Statement have been made, it is necessary to regenerate the SQL statement in order to be able to switch the screen back to the mode to visually build the SQL for the validation file.

Auto Generation and the Validation Properties Screen

Once a validation file has been set up on the *ODBC Validation Source* screen, the validation file can be accessed from the *Validation File Properties* screen like any other validation file. The validation properties screen has an extra feature in **IT*Script*Net**. The Auto-Generate drop-down box controls the type of auto-generation that is used for the validation file. There are three choices described below.



No Auto Generation

This option will disable the auto-generation settings. A validation file that was set-up originally as a text file will have the No Auto-Generation option selected. The Manual or On Every Upload option

can be selected at any time. Doing so will enable the **Edit** button and allow access to the *ODBC Validation Source* screen so that the auto-generation can be configured.

Auto Generate-Manual

If a validation file was created using the *ODBC Validation Source* screen, the Manual option will be already selected. The Manual option means that the validation file contains the additional information necessary so it can be auto-generated from an ODBC Data Source. However, the actual step of creating the validation file must be done manually by clicking on the **Generate Val File** button on the *ODBC Validation Source* screen. Clicking on the **Edit** button next to the Auto Generate options can access the *ODBC Validation Source* screen.

Auto Generate-On Every Upload

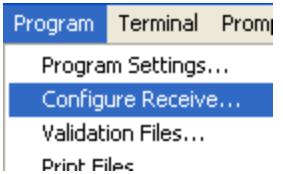
The On Every Upload option will cause the validation file to be generated every time the program is uploaded to a terminal. This option assures that the validation file is always up-to-date and is therefore the generally recommended option for auto-generating validation files. The Validation file will also get generated when the program is uploaded via the OMNI Server.

Auto Generate-Remote

The Remote option is a fourth choice that is only available in **IT*Script*Net OMNI.** The remote option will cause the validation file not to be generated until it is needed during remote data collection. A prompt or element that uses a remote validation file will cause the remote validation file to be generated on-demand when the prompt or element needs the validation file.

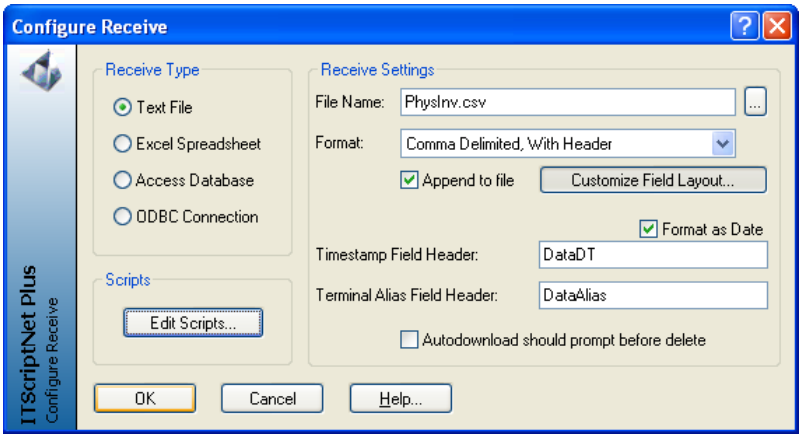
Configure Receive

The *Configure Receive* main menu item will display the screen that allows you to configure what **IT*Script*Net** will do with the collected data after it retrieves the data from the terminal. Click on Program on the menu bar and select Configure Receive.



Receive Type

IT*Script*Net supports four download formats: Text File, Microsoft Excel, Microsoft Access, and ODBC. The detailed download settings on the right side of the screen will change depending on the download format you select on the left side of the screen.



Text File

Use this option to receive your collected data into a text file. Click on the Text File option as the Receive Type on the left side of the screen to configure your data collection program to receive its data in text file format.

File Name

This field allows you to specify the name of the file in which you want your collected data to be stored. You may use the **'...'** (Browse) button to browse for an existing file, or simply type the name of a new file that you want to be created. If you specify a fully qualified path name, that path will be used when downloading your collected data file. If a file name only is specified, the file will be placed in the same directory as the ITB file when downloading.

Format

You can select between fixed-width text files or comma-delimited text files. You can further dictate whether the comma-delimited text file includes field headers in the first row of the file. If you choose to use a comma-delimited file and name the file with a .CSV extension, you will be able to open the file directly in Microsoft Excel. The data collected for all prompts in the data collection program will be included in the output file format in the order that they appear in the program. You can use the Customize Field Layout option to configure which fields to include in the data output and the order of the fields. See the Customize Field Layout section later in this chapter for more information.

Append To File

This option controls whether the collected data will be appended to the end of the data file if it already exists, or if the existing file will be deleted and replaced with a new file containing only the new data. Check the box to have your new data appended to the existing file. Uncheck the box to have a new file created each time. If the file does not exist, it will be created.

Timestamp Field Header

If you select Fixed Width or Comma-Delimited Without Headers, this field is disabled. If you select Comma-Delimited With Headers, this field specifies the name that will be given to the Timestamp field in the file. Each record collected in the terminal has a date and time stamp attached to it. When the PC receives the data, the timestamp is kept with the record. If you leave this field blank, the timestamp will not be saved in the output file.

Format As Date

This option is only available when using Comma-Delimited format. If checked, the date will be stored in a localized date format. If unchecked, the date will be in YYYYMMDDHHNNSS format where YYYY is the 4-digit year, MM is the month, DD is the day, HH is the hour, NN is the minute, and SS is the second.

Alias Field Header

If you select Fixed Width or Comma-Delimited Without Headers, this field is disabled. If you select Comma-Delimited With Headers, this field specifies the name that will be given to the Alias field in the file. Each record collected in the terminal has the terminal's Alias attached to it. When the PC receives the data, the alias is kept with the record. If you leave this field blank, the alias will not be saved in the output file. The Terminal Alias is set on the terminal's configuration screen.

Autodownload Prompt Before Delete

This option on the *Configure Receive* screen will control whether or not the data for the data collection program will be automatically deleted after being downloaded with the Autodownload Server application. There are several means of getting collected data from the terminals to the PC for receipt and processing. One of these methods is via the Autodownload Server. The Autodownload Server will allow collected data to be sent to the PC automatically when the user places the terminal in the cradle. The default behavior is to have the collected data be deleted after this automatic transfer to minimize user involvement in the process. When checked, this option will cause the terminal to prompt the user before deleting the collected data after transfer. In some cases this may be desirable so that there is a user involved to confirm that the data was transferred and processed as expected before the data is deleted from the terminal. The Autodownload Sever application has a dedicated section in the User Guide for further information.

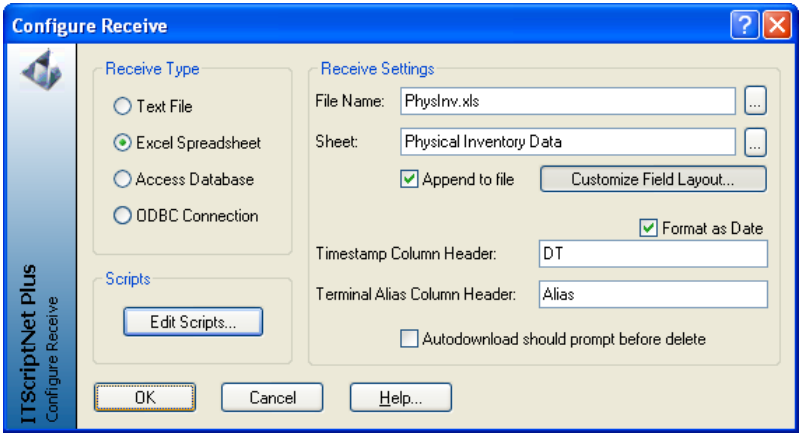
Customize Field Layout

This button is used to change the order in which fields will be placed in the output file, and is described in detail later in this document.

Excel Spreadsheet

Use this option to receive your collected data into a Microsoft Excel spreadsheet. Click on the Excel Spreadsheet option as the Receive Type on the left side of the screen to configure your data collection program to receive its data into Microsoft Excel. You must have Excel installed on your PC to be able to receive your collected data into an Excel file.

File Name



This field allows you to specify the name of the spreadsheet file in which you want your collected data to be stored. You may use the **'...'** button to browse for an existing spreadsheet, or simply type the name of a new spreadsheet that you want to be created. If you specify a fully qualified path name, that path will be used when downloading your collected data file. If a file name only is specified, the file will be placed in the same directory as the ITB file when downloading.

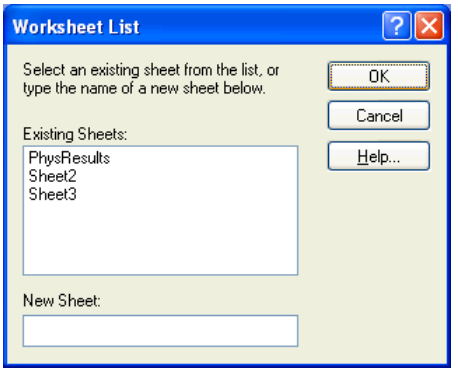
The data collected for all prompts in the data collection program will be included in the output file format in the order that they appear in the program. You can use the Customize Field Layout option to configure which fields to include in the data output and the order of the fields. See the Customize Field Layout section later in this chapter for more information.

Sheet Name

This field allows you to specify the name of the sheet within the spreadsheet file in which you want your collected data to be stored. You may type the name of an existing sheet or a new sheet that you want to be created. You may also use the **'...'** button to select an existing sheet from the spreadsheet using the Worksheet List.

Worksheet List

This screen displays a list of the sheets in the spreadsheet. You may select an existing sheet, or enter a new sheet name in the New Sheet line.



Special Sheet Names

There are three special codes you may use as sheet names. These codes cause the sheets to be named dynamically. The valid codes are:

CODE	OPERATION
$DATE$	Names the sheet using the current date. The format will be MM-DD-YY.
$TIME$	Names the sheet using the current time. The format will be HH'MM'SS
$DATETIME$	Names the sheet using the current date and time. The format will be MM-DD-YY HH'MM'SS

Append To File

This option controls whether the collected data will be appended to the end of the sheet if it already exists, or if the existing data will be overwritten and replaced with the new data. Check the box to have your new data appended to the existing file. Uncheck the box to have the new data overwrite the old data.

Timestamp Column Header

This field specifies the name that will be given to the Timestamp column in the spreadsheet. Each record collected in the terminal has a date and time stamp attached to it. When the PC receives the data, the timestamp is kept with the record. If you leave this field blank, the timestamp will not be saved in the output file.

Format As Date

If checked, the date will be stored in a localized date format. If unchecked, the date will be in YYYYMMDDHHNNSS format where YYYY is the 4-digit year, MM is the month, DD is the day, HH is the hour, NN is the minute, and SS is the second.

Alias Field

This field specifies the name that will be given to the Alias column in the sheet. Each record collected in the terminal has the terminal's Alias attached to it. When the PC receives the data, the alias is kept with the record. If you leave this field blank, the alias will not be saved in the output file. The Terminal Alias is set on the terminal's configuration screen.

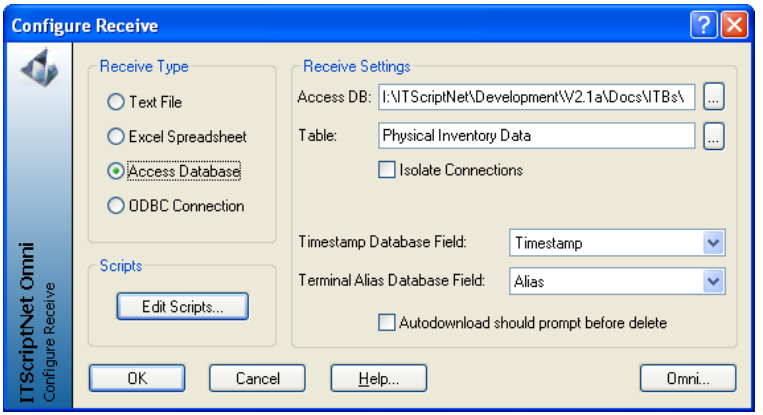
Autodownload Prompt Before Delete

This option on the *Configure Receive* screen will control whether or not the data for the data collection program will be automatically deleted after being downloaded with the Autodownload Server application. There are several means of getting collected data from the terminals to the PC for receipt and processing. One of these methods is via the Autodownload Server. The Autodownload Server will allow collected data to be sent to the PC automatically when the user places the terminal in the cradle. The default behavior is to have the collected data be deleted after this automatic transfer to minimize user involvement in the process. However, this option, when checked will cause the terminal to prompt the user before deleting the collected data after transfer. In some cases this may be desirable so that there is a user involved to confirm that the data was transferred and processed as expected before the data is deleted from the terminal. The Autodownload Sever application has a dedicated section in the User Guide for further information.

Customize Field Layout

This button is used to change the order in which fields will be placed in the output file, and is described in detail later in this document.

Access Database

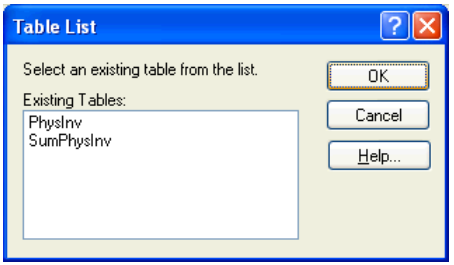


Use this option to receive your collected data into a Microsoft Access database. Click on the Access Database option as the Receive Type on the left side of the screen to configure your data collection program to receive its data into Microsoft Access. You must have ADO2.5 or higher installed on your PC to receive your data into an Access database. The setup program should have automatically installed ADO 2.5, but you can also download ADO from Microsoft's Universal Data Access web site at **http://www.microsoft.com/data** (download MDAC 2.5).

Access Database

This field allows you to specify the name of the database file in which you want your collected data to be stored. You may use the **'…'** button to browse for an existing database or type the name of the database in the field. The database must exist. The receive process will not create the database if it does not exist.

Table



This field allows you to specify the name of the table within the database file in which you want your collected data to be stored. You may type the name of an existing table or use the **'…'** button to select an existing table from the database using the Table List. You must specify an existing table. The table will not be created if it does not exist.

Isolate Connections

This checkbox allows you to specify that you want updates to the Access Database to be isolated, so that only one download process can update the database at a time. This option is available in the **IT*Script*Net OMNI** edition only.

Timestamp Field

This field specifies the name that will be given to the Timestamp field in the table. Each record collected in the terminal has a date and time stamp attached to it. When the PC receives the data the timestamp is kept with the record. You may specify the name of an existing field or select <Ignore> if you do not want the timestamp data to be saved.

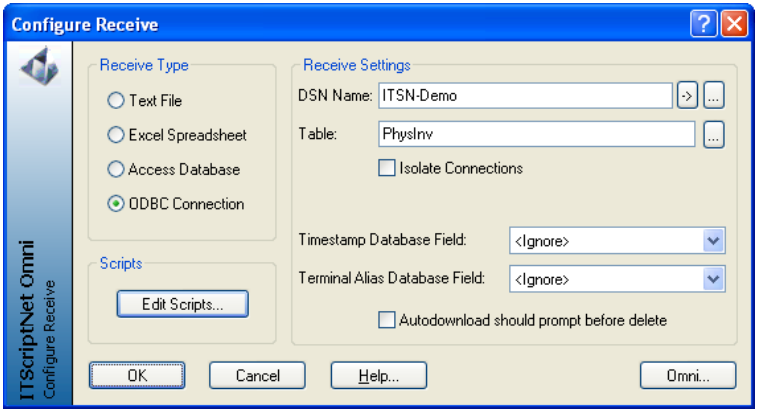
The download process will automatically detect the field type in the database, and adjust the data accordingly. If your field is a Text type, the field must be a 14-character text field. The timestamp data will be in YYYYMMDDHHNNSS format where YYYY is the 4-digit year, MM is the month, DD is the day, HH is the hour, NN is the minute, and SS is the second. If your field is a Date/Time type, the data will be converted into a Date type and assigned to the field.

Alias Field

This field specifies the name that will be given to the Alias field in the table. Each record collected in the terminal has the terminal's Alias attached to it. When the PC receives the data, the alias is kept with the record. You may specify the name of an existing field or select <Ignore> if you do not want the alias data to be saved. The Terminal Alias is set on the terminal's configuration screen.

Autodownload Prompt Before Delete

This option on the *Configure Receive* screen will control whether or not the data for the data collection program will be automatically deleted after being downloaded with the Autodownload Server application. There are several means of getting collected data from the terminals to the PC for receipt and processing. One of these methods is via the Autodownload Server. The Autodownload Server will allow collected data to be sent to the PC automatically when the user places the terminal in the cradle. The default behavior is to have the collected data be deleted after this automatic transfer to minimize user involvement in the process. However, this option, when checked will cause the terminal to prompt the user before deleting the collected data after transfer. In some cases this may be desirable so that there is a user involved to confirm that the data was transferred and processed as expected before the data is deleted from the terminal. The Autodownload Sever application has a dedicated section in the User Guide for further information.



ODBC Database

Use this option to receive your collected data into a database with an ODBC data source. Click on the ODBC Connection option as the Receive Type on the left side of the screen to configure your data collection program to receive its data into ODBC. This is an advanced topic. If you are unsure how to work with ODBC data sources, contact your system administrator. You must have ADO2.5 or higher installed on your PC to receive your data into a database with ODBC. The setup program should have automatically installed ADO 2.5, but you can also download ADO from Microsoft's Universal Data Access web site at **http://www.microsoft.com/data** (download MDAC 2.5).

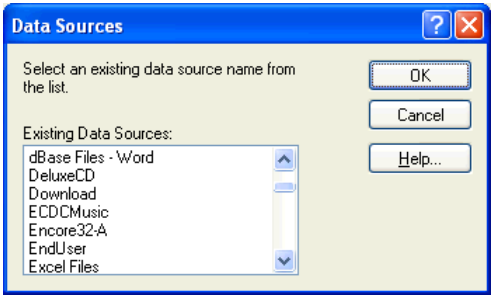
DSN Name

This field allows you to specify the name of the ODBC data Source in which you want your collected data to be stored.

You may use the **'...'** button to browse for an existing

data source, or type the name of the data source in the field.

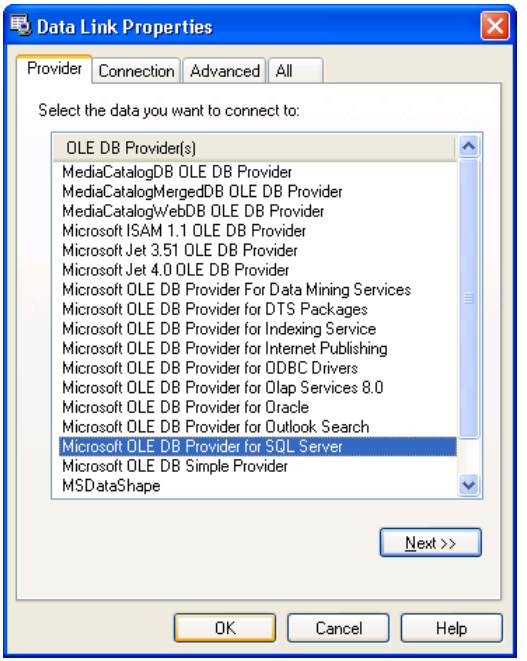
If you specify a data source name (DSN), the data source must exist. The receive process will not create the data source if it does not exist.



Connection String Data Link

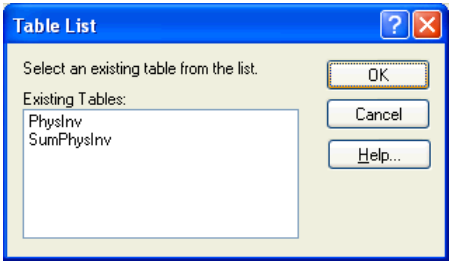
You can build a connection string for **IT*Script*Net** to use without first creating a DSN. Use the Data Link button to bring up the Data Link Properties screen

and follow the steps of choosing a Provider and setting Connection properties to build a connection string. The connection string contains all the information necessary for **IT*Script*Net** to access the ODBC Connection without a predefined DSN.



Table

This field allows you to specify the name of the table within the database file in which you want your collected data to be stored. You may type the name of an existing table or use the **'...'** button to select an existing table from the data source using the Table List. You must specify an existing table. The table will not be created if it does not exist.



Isolate Connections

This checkbox allows you to specify that you want updates to the Database to be isolated, so that only one download process can update the database at a time. This option is available in the **IT*Script*Net OMNI** edition only.

Timestamp Field

This field specifies the name that will be given to the Timestamp field in the table. Each record collected in the terminal has a date and time stamp attached to it. When the PC receives the data, the timestamp is kept with the record. You may specify the name of an existing field or select <Ignore> if you do not want the timestamp data to be saved.

The download process will automatically detect the field type in the database, and adjust the data accordingly. If your field is a Text type, the field must be a 14-character text field. The timestamp data will be in YYYYMMDDHHNNSS format where YYYY is the 4-digit year, MM is the month, DD is the day, HH is the hour, NN is the minute, and SS is the second. If your field is a Date/Time type, the data will be converted into a Date type and assigned to the field.

Alias Field

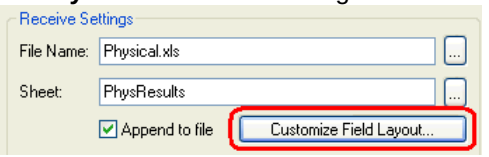
This field specifies the name that will be given to the Alias field in the table. Each record collected in the terminal has the terminal's Alias attached to it. When the PC receives the data, the alias is kept with the record. You may specify the name of an existing field or select <Ignore> if you do not want the alias data to be saved. The Terminal Alias is set on the terminal's configuration screen.

Autodownload Prompt Before Delete

This option on the *Configure Receive* screen will control whether or not the data for the data collection program will be automatically deleted after being downloaded with the Autodownload Server application. The Autodownload Server will allow collected data to be sent to the PC automatically when the user places the terminal in the cradle. The default behavior is to have the collected data be deleted after this automatic transfer to minimize user involvement in the process. When checked, this option will cause the terminal to prompt the user before deleting the collected data after transfer. In some cases this may be desirable so that there is a user involved to confirm that the data was transferred and processed as expected before the data is deleted from the terminal. The Autodownload Sever application has a dedicated section in the User Guide for further information.

Customize Field Layout

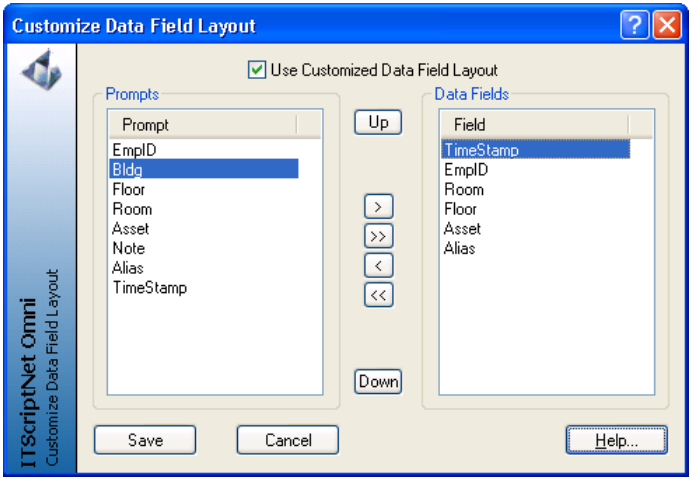
The *Configure Receive* screen includes the **Customize Field Layout…** button. Clicking this button displays the *Customize Data Field Layout* Screen. This screen is where a text or Excel output file can be customized. Normally, the order of the fields in a text or Excel output file are in the same order as the prompts in the program and the responses for all prompts appear in the data output file. This screen allows the order of these fields to be customized so that the fields can match up to a specific desired format. The output data file can also be customized to omit the fields for selected prompts.



The top of the screen contains a checkbox that determines whether or not customization is active.

The left side of the screen contains a list of the prompts and elements in the data collection program. The terminal Alias and Timestamp are also in the list because the Alias and Timestamp are captured for each record.

The right side of the screen is the list of data fields to appear in the output file (either a text file or Excel file depending on the settings chosen on the *Configure Receive* screen). You can use the arrow keys in the middle of the screen to add or remove data fields from the right-hand list. You can use the **Up** and **Down** buttons to manipulate the order of the fields in the list. Only the fields in the right-hand list will be included in the output file if the customized data checkbox is checked. The order of the fields in the output data file will match the order of the fields in the right-hand list.



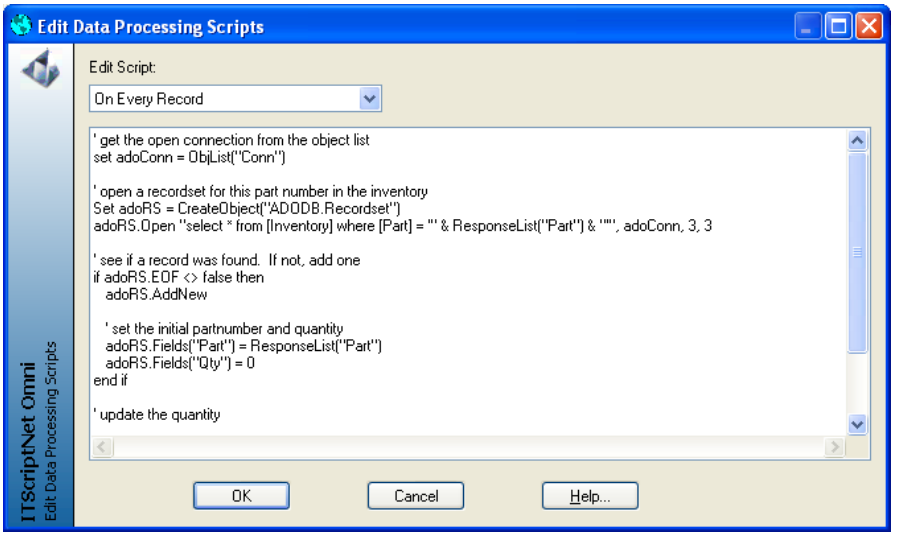
In the example shown, there are many fields that were not specified to be included in the output file. Also, the order of the fields has been changed so that the first element in the output file will be the Timestamp. At any time, if the checkbox is unchecked, the data output file will revert to the normal default output file.

Data Processing Scripts

The *Configure Receive* screen includes the **Edit Scripts** button. Clicking this button displays the *Edit Data Processing Scripts* screen. This is an area where VBScripts can be defined for the data collection program's data processing. These VBScripts run to process the collected data as the PC receives the data. These scripts are optional.

Types of Data Processing Scripts

There are several scripts that can be used. Each runs at a different time during the download process. To select the script to edit, use the *Edit Script* dropdown box. The script code can then be edited in the editing window below.



Before Uploading

This script runs once before any files are uploaded to the terminal from the PC. This script could be use to copy support or validation files, start a validation file generation process in Host software, or any other task that needs to be accomplished before uploading to the terminal. Note that this script runs before any automatic Validation File or Index File generation.

After Uploading

This script runs once after all of the files have been uploaded to the Terminal from the PC. This script could be used to clean up any files that were sent to the terminal, or any other task that should happen after the Upload process.

Before Downloading

This script runs once before the data is downloaded from the Terminal to the PC. This script could be used to prepare a download directory, or perform any other task that should be done before the data is downloaded from the Terminal.

Before Processing Data

This script runs once after the collected data is received, but before processing. This would typically be a script that would open connections to existing databases or perform other set-up tasks.

For Each Record

This script runs for each record before the native **IT*Script*Net** processing occurs. This script usually contains the bulk of the processing logic. The logic of the script could be complex enough to update several tables based on the data content or could tap into existing business logic.

After Processing Data

This script runs once after all records have been processed and often is used to send messages to the user, close database connections, or perform any other final tasks before processing is considered complete.

Accessing the Collected Data from the Scripts

The collected data is available to the **For Each Record** script in a pre-defined collection named **ResponseList**, keyed by the Prompt Name for single prompts and by "Prompt.Element" for Multi-Prompts. For example, if you have a single prompt named "Prompt1", you can access the data collected for that prompt by using **ResponseList("Prompt1**"). If you have a Multi-Prompt named "Prompt2" that has an element named "ElementA", you can access the data collected for that element by using **ResponseList("Prompt2.ElementA**"). You may make changes to the collected data and save those changes back to the **ResponseList**. Changes made in this way will be saved in the collected data file. Any collected data fields that are not modified will be saved as they were collected. See the Script Sample for an example.

Note that the ResponseList is only available in the *For Each Record* script.

Passing Objects Between Scripts

You can pass objects (such as Database Connections, Recordsets, or COM Objects) between the scripts using the predefined collection named **ObjList**. This collection is keyed by a name you specify. For example, you could store a database connection as **ObjList("Connection").** The **ObjList** retains the objects placed into it from one script to the next. See the Script Sample for an example.

Note that the ObjList is only available in the *Before Processing Data, For Each Record,* and *After Processing Data* scripts.

Skipping Records

You can force a record to be skipped and not placed in the collected data file. To do this, set the pre-defined variable named **ProcessRecord = 0**. This will cause the data record to be skipped as though it had not been collected.

Sample Script

The following is a sample script using the Download scripts:

Before Processing Script:

```
set adoConn = CreateObject("ADODB.Connection")
adoConn.Open "DSN=Function"
Set adoRS = CreateObject("ADODB.Recordset")
adoRS.Open "select * from function where ID = -1", adoConn, 3, 3
set ObjList("Conn") = adoConn
set ObjList("RS") = adoRS
```

The script creates and opens a database connection an empty recordset, then stores them in the ObjList so they will be available to the other scripts.

For Each Record Script:

```
set adoRS = ObjList("RS")
adoRS.AddNew
adoRS.Fields("PrePrompt") = ResponseList("preprompt")
strDate = trim(ResponseList("timestamp"))
adoRS.Fields("RealDate") = mid(strDate,5,2) & "/" & mid(strDate,7,2) &
"/" & left(strDate,4) & " " & mid(strDate, 9, 2) & ":" & mid(strDate,
11,2) & ":" & mid(strDate,13,2)
ResponseList("Alias") = "NewTerm"
adoRS.Update
```

This script is run once for each record. The recordset is retrieved from the ObjList, then the data for the fields is read from the ResponseList. In this sample, the timestamp field is reformatted and the alias is changed. Note that you can change the data and assign it back to the ResponseList.

The changed data will be processed as though the terminal operator had collected it. The other fields in ResponseList are saved unchanged.

After Processing Script:

```
set adoConn = ObjList("Conn")
set adoRS = ObjList("RS")
adoRS.Close
adoConn.Close
set adoRS = nothing
set adoConn = nothing
```

This script cleans up by closing the recordset and database connection.

File Menu

New / New Program Wizard

The first two menu items under the File main menu item will create new data collection programs. The *New* menu item will create a blank program with one prompt. You can also create a new program by clicking on the *New Program Wizard* menu item, which will walk you through the process of quickly creating a program using the New Program Wizard. The wizard is a great way to create a simple program or to form the basis for a more complicated program. It is also a good way for users who are new to **IT*Script*Net** to become familiar with prompts and their properties.

Open

The *Open* menu item will allow you to open an **IT*Script*Net** program. A standard Windows file open screen will assist you in browsing for the file you wish to open. **IT*Script*Net** programs use an .ITB file extension.

Save / Save As

The *Save* menu item will save the **IT*Script*Net** program that you are working on to a file. The file will be saved to the drive, folder, and file name that you opened. The *Save As…* will open the standard Windows save screen, which will allow you to save the program to any drive or file name that you wish.

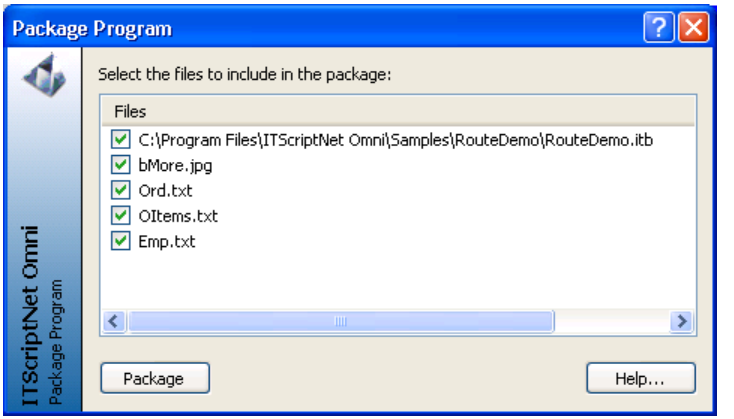
Save As Batch

This option allows you to save the current program down to **IT*Script*Net Batch** format. The program must be saved under another filename. Any features that you used that are not available in Batch will either be removed or will not operate in the Batch program.

Package Program

This option is used to create a ZIP file containing the data collection program and all of the support and validation files. This is useful when you need to send a complete set of files to another location. You can package the program and email it or burn it to a CD-ROM.

When you load the screen, the list shows the Data collection program and all support or validation files.



Each file has a checkmark indicating that it will be included in the package. You can uncheck any files that you do not want included. Click Package to specify the filename and create the package.

Print / Print Preview / Print Setup

The *Print* menu item will print your program's prompts in flowchart form exactly as they are displayed in the left-hand side of the main application window. The printout will also include the detailed property information for the selected prompt. **IT*Script*Net** also includes the standard Windows Print Preview feature and the *Print Setup* screen.

Recently Used Files

IT*Script*Net will track your most recently used **IT*Script*Net** programs (.ITB files) and will list them under the *File* menu. You can select one of your recent files from the menu, and the program files will be opened just as if you had used the *Open* menu item and browsed for the file. The recently used file list is a convenient shortcut for opening your files quickly.

Exit

Selecting the *Exit* menu item will close **IT*Script*Net**. Make sure any changes have been saved before exiting.

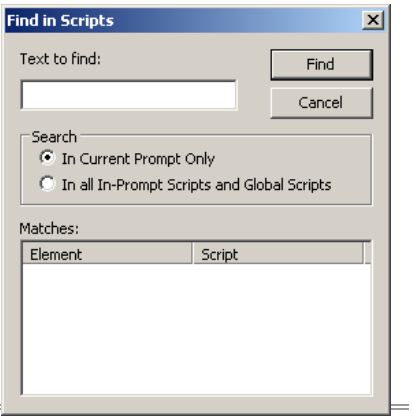
Edit Menu

Cut / Copy / Paste

These menu items will allow you to cut or copy and then paste prompts as described in the Prompting Sequence section of this User Guide.

Undo

IT*Script*Net supports limited Undo functionality when editing Multi-prompts. This will allow you to undo settings changes, movement on screen, delete, etc. Note that not all operations can be undone, and moving from one prompt to another or using a program-level screen resets the last Undo point.



Find

The *Find* menu item allows you to search your in-prompt scripts for a specific word or phrase. You can choose to search in the current prompt only, or search in all prompts.

If the word or phrase is found, the matching scripts will be displayed in the *Matches* list at the bottom of the dialog box. You can double-click one of these entries to edit the script.

Properties

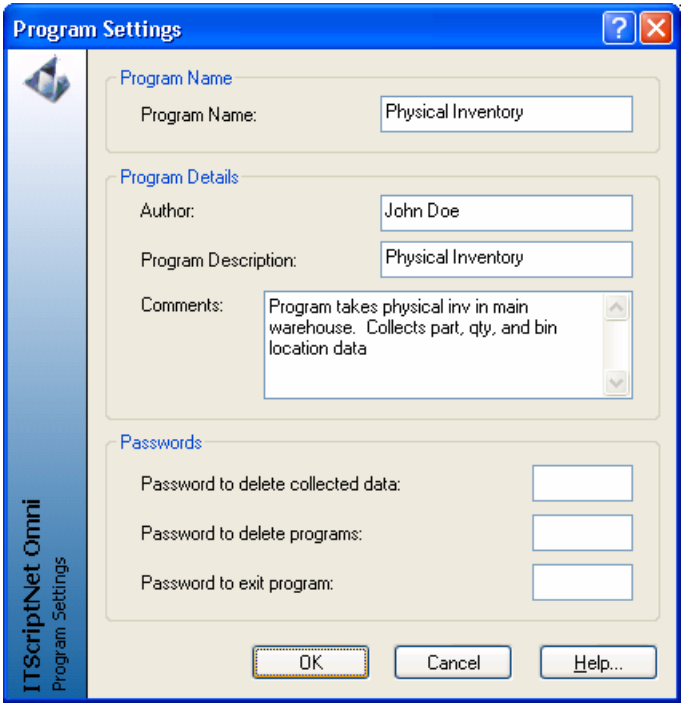
The *Properties* menu item will display the *Prompt Settings* screen for the selected prompt. Refer to the Prompt Settings section of this User Guide for a complete description of each prompt setting available in **IT*Script*Net**.

Program Menu

The *Program* menu has options to allow you to control options global to the entire program.

Program Settings

The *Program Settings* screen allows you to set your data collection program's name, additional information, and associated passwords.



Program Name

This field allows you to set the name that will be displayed on the terminal when selecting a data collection program from the list of programs on the terminal. If you leave this field blank, it will be defaulted to the same name as the program file.

Program Details

The Program Details allow you to enter some additional information about your program. These fields are all optional.

Author

This program detail allows you to record the author of the data collection program.

Program Description

The Program Description field is used to store a short description of the data collection program.

Comments

The comments field is a place to record notes about the data collection program to describe its purpose, revision history, etc.

Passwords

There are several passwords that can be assigned to a data collection program to add security to sensitive program functions. None of the passwords are required. They are optional security measures.

Password to Delete Collected Data

You may specify a password that must be entered on the terminal in order to delete the collected data file. This password is only effective when selecting the *Delete Data* menu option on the terminal. The data may be deleted after using the *Send Data to PC* option without the password. If no password is specified, any user can delete the data.

Password to Delete Programs

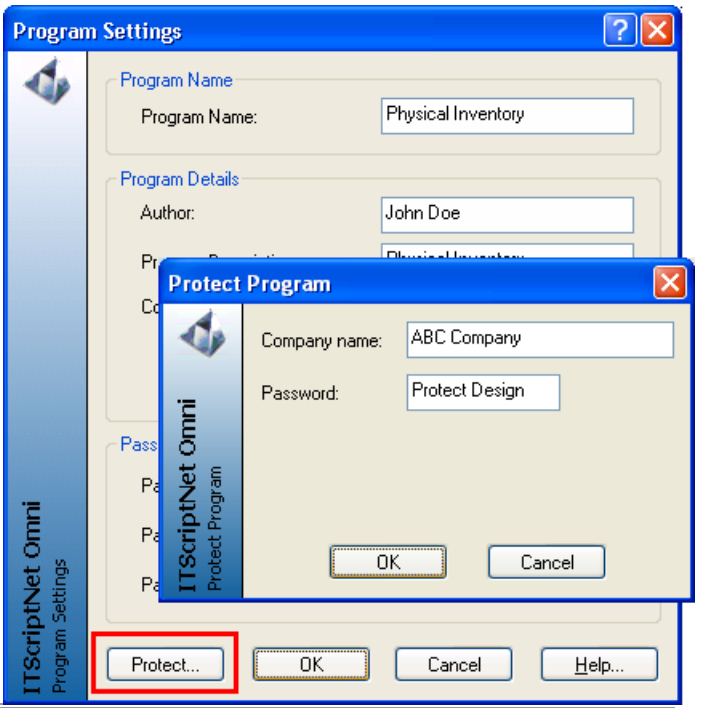
You may specify a password that must be entered on the terminal in order to delete a program using the *Delete Program* menu. If no password is specified, any user can delete the program. Because **IT*Script*Net** supports multiple data collection programs on a given terminal, the operator must specifically delete a program from the terminal if it is no longer used. Deleting the unused data collection program from the terminal will keep the terminal's resources free for other data collection programs or data files.

Password to Exit Program

The last password option is that you can specify a password that must be entered on the terminal in order to exit this program's data collection. This means that a user that does not have this password will not be able to get back to the **IT*Script*Net** main menu.

Protect Password

IT*Script*Net has a password protection feature available in the Plus and OMNI editions. Click on the **Protect** button on the *Program Settings* screen to bring up the *Protect Program* screen. Enter the company name and a password. This password will protect the data collection program from unauthorized changes. Anyone attempting to open an **IT*Script*Net** file with a protect password will be required to enter the correct password in order to open the program for design.



Configure Receive

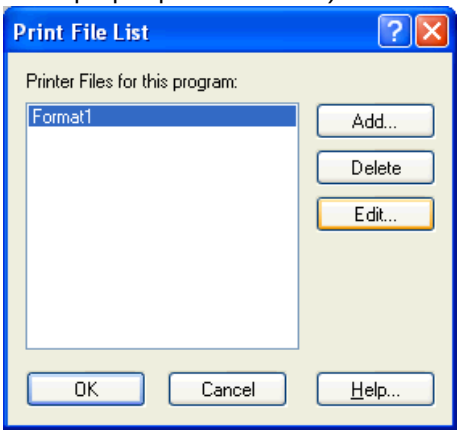
This menu item will display the screen that allows you to configure what **IT*Script*Net** will do with the collected data after it retrieves the data from the terminal. A complete description of this menu item's screen can be found in the *Configure Receive* section of this User Guide.

Validation Files

An **IT*Script*Net** data collection program can use multiple validation files. Validation files are used so that responses to prompts can be validated against a set of data to determine if the response is acceptable. Use of validation files helps assure that the data collected is accurate. Each validation file to be used to validate data for a prompt within a program must be defined for the program on the *Validation Files* screen. A complete description of this menu item's screen can be found in this User Guide in the *Configuring Validation Files* section.

Print Files

IT*Script*Net Plus and **IT*Script*Net OMNI** allow printing to IrDA, RF, Serial or Bluetooth printers from portable terminals that can support printing (the terminals have the proper ports or radios). There are three methods to print to a printer, one for each of the Print functions described in the Print/Other functions section of the function reference. One of the methods utilizes Print Files. The *Print Files* menu option creates and manages print files.



Selecting the *Print Files* menu item displays the *Print File List* screen. Any print files that have been configured will be displayed in the list. Print files included in a data collection program will be uploaded to a terminal with the program, validation files, and other files necessary for the program to function on the terminal.

Add

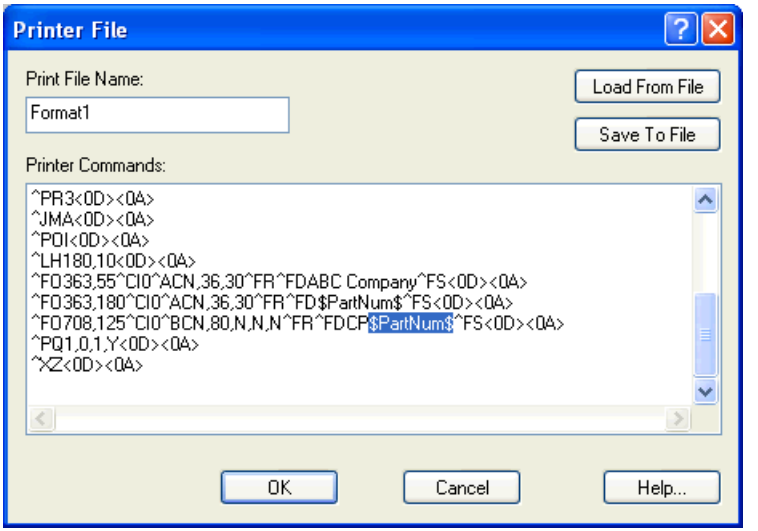
Click the **Add** button to add a new print file to the program and bring up the *Printer File* screen, detailed in the Edit section below.

Delete

Click the **Delete** button to remove the selected print file from the list.

Edit

Click the **Edit** button to bring up the *Printer File* screen.



Print File Name

This is the name of the print file. It is not a disk file name, but a logical name that is used by the Printing functions to reference the print file.

Printer Commands

The area allows you to enter your actual printer commands. This is the raw data that will be sent to the printer.

Non-printable ASCII characters can be encoded by entering them in the format <HH> where HH is the 2-digit Hexadecimal code for the character. For example, a carriage return [CR] is represented by <0D> and a line feed [LF] by <0A>. Program variables can be substituted into the printer data by including $, #, and @ variables in the data just like in the Text display fields. In the example shown, the $PartNum$ variable has been inserted into the printer commands. During printing on the terminal, **IT*Script*Net** will insert the value of the PartNum prompt into the data stream being sent to the printer. If you needed to substitute for a response to an element on a multi-prompt, you would use the $Prompt.Element$ variable naming convention.

Load From File

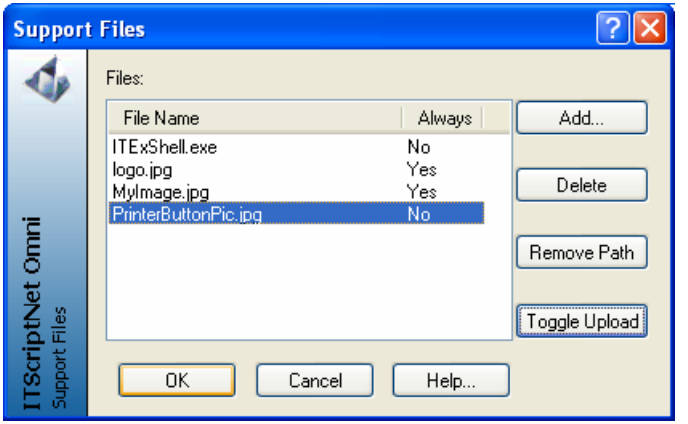
The **Load From File** button loads a printer command stream from a file. For example, a form could be designed with a label design package and then printed to a file. The **Load From File** button could then be used to browse for the print file containing the printer data stream. The data stream is loaded into the *Printer File* screen so it can be modified with variable substitution.

Save To File

The **Save To File** button saves the modified printer commands to a file on disk.

Support Files

The *Support Files* screen lists additional files that will be sent to the Portable Terminal along with the program and validation files. This allows Image files, Shell programs, or other required files to be sent with the **IT*Script*Net** program to the terminal.



File List

The File List shows the full path to any supporting files that will be sent to the terminal any time the data collection program is sent to the terminal. Note: If you are using a DOS-based terminal make sure that all supporting file names are Short names that fit the 8.3 naming convention.

Add and Delete

Click the **Add** button to add a new supporting file to the list. When you click the **Add** button you will be able to browse for the support file that you wish to add. Click the **Delete** button to delete the selected supporting file from the list.

Remove Path

The **Remove Path** button will strip the path from the file name. If the path is removed, **IT*Script*Net** must be able to locate the supporting file in the same directory as the data collection program (.itb file). It is often useful to remove the paths from the support files and keep the supporting files in the same directory as the .itb file so that if the files are moved they can be moved together without the need to reset the paths on each supporting file. The capability to have no paths specified also makes the solution easier to deploy.

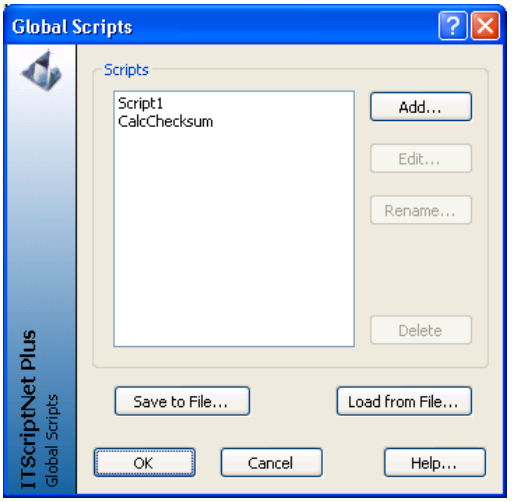
Toggle Upload

The **Toggle Upload** button will change the value of the 'Always' column for the selected support file in the list. If the value in the 'Always' column is 'Yes' then the toggle upload button will cause it to say 'No' and vice versa. When the data collection program is uploaded to the terminal, the support files normally get sent to the terminal also. If the 'Always' column is set to 'Yes' then the support file will always be sent with the data collection program. If the column is set to 'No', then the user's options selected on the upload screen will determine if the file gets sent. The purpose behind this feature is to minimize upload time for programs that have many or large supporting files. Often support files that are images or .WAV files would not change, but the program may be frequently uploaded in order to maintain current dynamic validation files. The toggle upload feature will allow the user to minimize the upload time by sending only those support files marked as 'Always'.

Global Script Editor

IT*Script*Net supports creating global scripts. These are any number of lines of script code that can be called from another script at any time. You can use these to encapsulate pieces of code that you want to call from several places throughout the program without cutting and pasting the same code over and over. Global Scripts are supported on PocketPC and Windows CE terminals, and the PC Client. DOS-Based terminals do not support global scripts.

You create a Global Script using the Global Script editor, and call it using the *GlobalScript* and *GlobalScriptFile* functions.



Add

Click Add to create a new global script. You will be prompted to enter the name of the global script, then the standard Script Editor is used to enter the script code. The script name will be added to the list.

Edit

This button is used to edit the currently selected global script code.

Rename

This button allows you to rename the currently selected global script.

Save To File

This button allows to you save the entire set of Global Scripts to a file. These scripts can then be shared among a number of programs using the *GlobalScriptFile* function.

Load from File

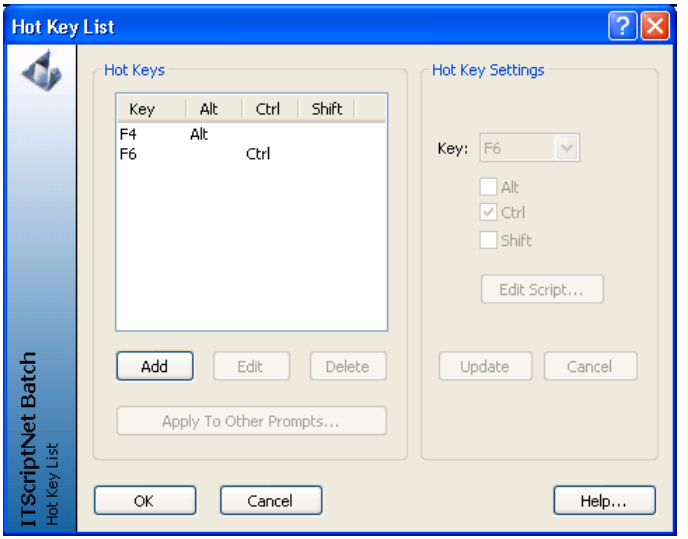
This button allows you to load a set of global scripts that were previously saved using the Save to File button. Any existing scripts will be deleted and replaced with the new scripts in the file.

HotKey Editor

IT*Script*Net supports executing a script in response to a Hotkey press. This key can be in combination with the Ctrl, Alt, or Shift keys. These hotkeys are supported on the WindowsCE, PocketPC terminals, and the PC Client only. DOS-based terminals do not support Hot Keys.

Hotkeys can be configured in two places: At Program Level, and at Prompt Level. When a hotkey is pressed, the client will check the Prompt settings first, and if the hotkey is not configured for that prompt, the Program configuration will be checked. This allows you to create Program level hotkeys that work on every prompt, but override them on a specific prompt if necessary.

The Hot Key List screen is used to configure the hotkeys for the program or a prompt. A Hotkey can be any Function Key from F1 to F24, or a Letter or Number key in combination with the Ctrl, Alt, or Shift keys. Function keys can be used as Hot Keys with or without Alt or Ctrl, but Letter and Number keys can only be Hot Keys in conjunction with Alt or Ctrl, and can not be used as hotkeys by themselves.



Add

Press ADD to Add a hotkey to the list. This activates the KEY dropdown box, the Alt, Ctrl, and Shift checkboxes, and the Edit Script button. Select the desired Function key and options, and press Edit Script to enter the script to be executed when the Hotkey is pressed. Press Update to save the hotkey into the list, or Cancel to abort the edit.

Edit

Select a hotkey in the list and press Edit. This activates the KEY dropdown box, the Alt, Ctrl, and Shift checkboxes, and the Edit Script button. Make whatever changes you want, and press Update to save. You can also press Cancel to abort your edit.

Delete

Select a hotkey and press Delete to remove the hotkey from the list.

Apply To Other Prompts.

This button will create a copy of the currently selected hotkey on each prompt. You can use this option when you want to have similar processing on each prompt, but do not want to use a Program level hotkey.

Key

Select the key to be used as the Hot Key. Note that Letter and Number keys also require that Alt or Ctrl be selected, but Function Keys do not.

ALT / CTRL / SHIFT

Select these check boxes to specify any modifiers to the Hot Key.

Edit Script

While configuring a Hot Key, you can use this button to write the script to be executed when the hotkey is pressed. This brings up the standard In-Prompt Script Editor screen.

Update

Press this button to save your changes to the Hot Key back into the list.

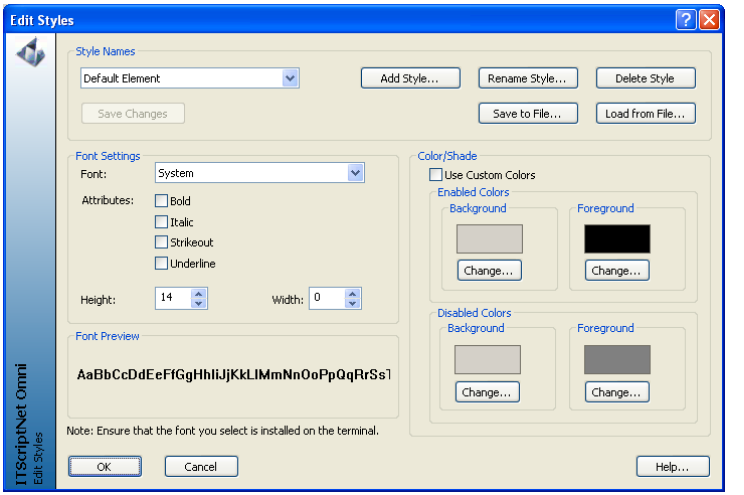
Cancel

This button cancels your changes and exits the Edit mode.

Style Editor

IT*Script*Net supports applying visual styles to multiprompt elements. This allows the program designer to easily change the font and color of objects throughout the program in a single step.

Style Names



Each style has a name which is used to reference the style when it is applied to the element. Select the name of the style that you want to edit from the Style Names drop down list.

Add Style

Use this button to create a new style. Enter the name you want to use for the new style.

Rename Style

Use this button to change the name of an existing style. Any elements that used the old style will no longer have a style applied, but will keep the font and color settings that they had last.

Delete Style

Use this button to delete the selected style. Any elements that used the old style will no longer have a style applied, but will keep the font and color settings that they had last.

Save to File

Use this button to save your style definitions to an external file. You can use this external file to transfer your styles from one ITB to another.

Load from File

Use this button to load your style definitions from an external file. You can use this external file to transfer your styles from one ITB to another.

Save Changes

Use this button to save the changes you made to a style before changing to another style.

Font Settings

Use these settings to select the font and attributes to use for the style. Make sure the font you select is available on the data collection device that you use.

Color/Shade

Use these settings to control the colors to use for the style. Note that not all color selections are supported by all elements. Check the individual element documentation for more information.

Font Preview

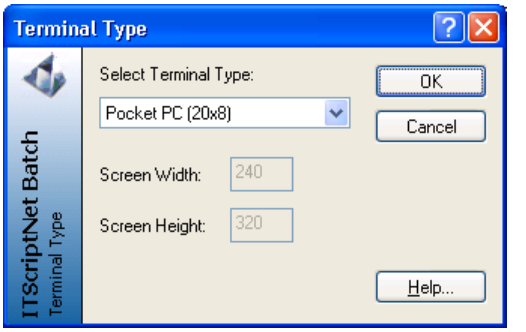
This area displays a sample of the text using the selected font. Use this to preview what the font will look like on the data collection device.

Terminal Menu

The *Terminal* menu contains the features that control how the program interfaces to the actual data collection terminals.

Select Terminal

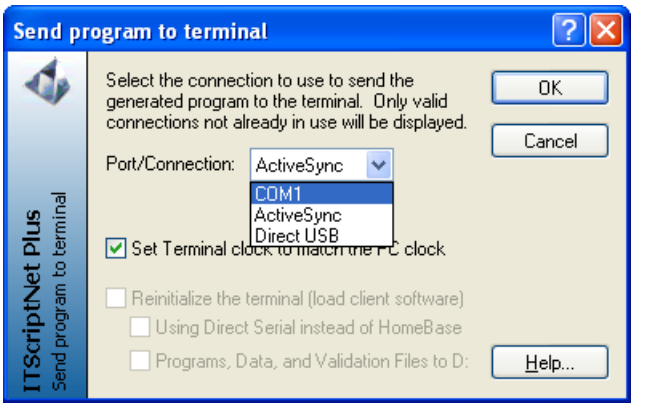
The *Select Terminal* menu item will bring up the *Terminal Type* screen. This screen sets the data collection terminal for which you are designing the program. The supported terminals are listed in the drop-down selection list. Select the type of terminal that you have and click OK to configure your program to use the selected terminal. Selecting the type of data collection terminal lets **IT*Script*Net** know how to configure the program for your terminal including setting the display for the proper terminal screen size and whether or not to enable image capture features.



If your device supports a variable screen size, for example the PC Client, the Screen Width and Screen Height controls will be active, and will allow you to change the size of the data collection program main window. If your terminal has a fixed screen size, then these controls will be set to their default values and disabled.

Send Program to Terminal

When you have finished designing your data collection solution, it is time to send your program to the portable terminal. Select the *Send Program to Terminal* menu item.



Port/Connection

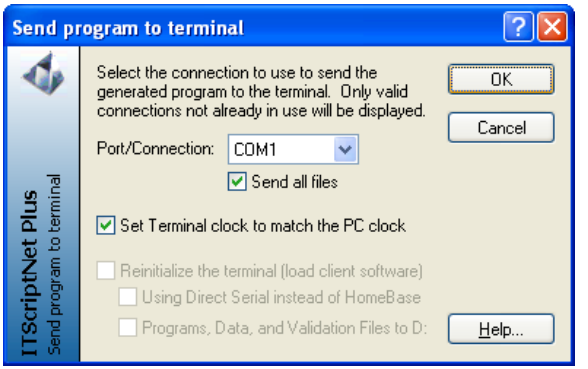
You must specify which port or connection to use to connect to your terminal from the options available to you, which will depend on your terminal type, the available ports on your PC, and which **IT*Script*Net** edition you are using. The available ports or connections will be displayed in the drop-

down list. The COM and USB ports are physical ports on the PC whereas the Microsoft ActiveSync Connection will use an existing ActiveSync link to a terminal. ActiveSync itself has options that specify the port within ActiveSync. The RF connection option will use a wireless LAN connection to the terminal and is only an option available in the **IT*Script*Net OMNI** package.

Regardless of the port/connection selected, the terminal must be configured to match. Refer to the terminal section for your terminal for more details on configuring the terminal for communication. Note that for serial communications, the baud rate and communication settings are not configurable – they will be set to the factory default settings that come set in the terminal.

Send All Files

By default, the Send All Files option will be checked so that the upload will send the data collection program, its validation files, and all supporting files to the terminal. However, not all support files always need to be sent. The *Support Files* screen allows the program designer to designate which support files should always be uploaded with the data collection program.



Set Terminal Clock

There is an option to send the date and time to the client terminal. The clock option defaults to 'on' so that the terminal's clock is kept up-to-date. This is important for data collection since the date and time of each record is recorded. This option can be unchecked to not send the date and time, if desired.

Reinitialize (Load Terminal Client)

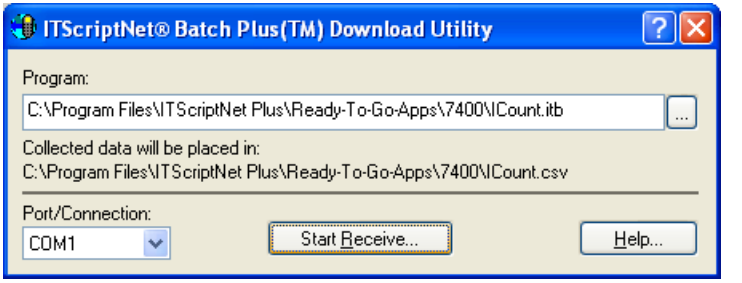
When sending your program to an **IT*Script*Net** supported DOS terminal, you can enable the option to reinitialize the terminal in addition to sending your data collection program. Reinitializing the terminal will send the **IT*Script*Net** Client software to the terminal in addition to your data collection program. **IT*Script*Net** Client software is a program that runs on the terminal and works with your data collection program to prompt the user, collect responses, and collect and store the data. You must enable the 'Reinitialize Terminal' option the first time that you send your program to the terminal in order to load the **IT*Script*Net** Client Software. After the first time, it will generally not be necessary to reinitialize the terminal unless you need to install an upgrade for the **IT*Script*Net** Client Software. Only the DOS terminals have the Reinitialize option. The Windows CE-based devices and the Pocket PC devices require that you load the client software through ActiveSync, so it is not an option when sending data to those terminals. Please see the section in this User Guide for your terminal for more specific information.

OK

Click on the **OK** button to begin the process of sending your program. You will need to follow the directions on your screen, as the upload procedure is specific to each type of terminal. For more help, please see the section in this User Guide for your specific type of terminal.

Receive a File from the Terminal

After data has been collected, you will need to retrieve the data from the terminal. Select the *Receive a File from the Terminal* menu item from the *Terminal* main menu item.



Program

You must specify the name of the program whose data you wish to retrieve. This will be filled in automatically with the name of the current program. With **IT*Script*Net** you can have several data collection programs loaded into the terminal at the same time, so it is necessary to specify from which of the data collection programs you wish to retrieve data. The receive screen will display the location where it will place the collected data file.

Port/Connection

You must also specify which port or connection to use to connect to your terminal from the options available to you. The available ports or connections will be displayed in the drop-down list. The options available will depend on your terminal type and the available ports on your PC. The COM and USB ports are physical ports on the PC whereas the Microsoft ActiveSync Connection will use an existing ActiveSync link to a terminal. ActiveSync itself has options that specify the port within ActiveSync. The RF connection option will use a wireless LAN connection to the terminal and therefore is only an option available in the **IT*Script*Net OMNI** package. Regardless of the port/connection selected, the terminal must be configured to match. Refer to the terminal section for your terminal for more details on configuring the terminal for communication.

Start Receive

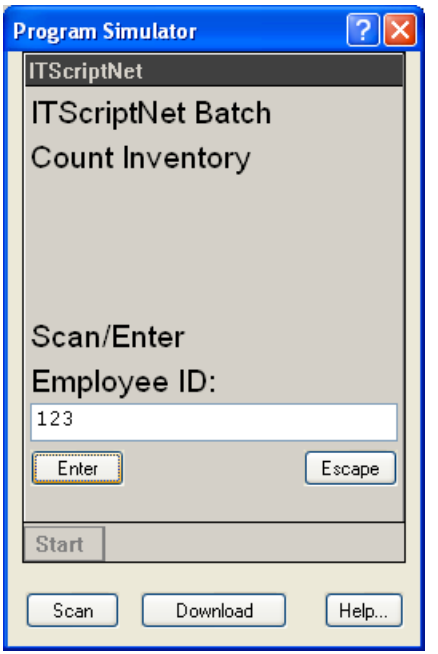
When you are ready, click the **Start Receive…** button to begin the process of retrieving your data. You will need to follow the directions on your screen, as the download procedure is unique to each kind of terminal. For more help, please see the section in this User Guide for your specific type of terminal.

Debug

The *Script Debugger* screen is very useful for troubleshooting complex In-Prompt scripts and/or complicated data collection programs with many In-Prompt scripts. Please refer to the *Debugging In-Prompt Scripts* section of this User Guide for more information about the Debug feature.

Simulator

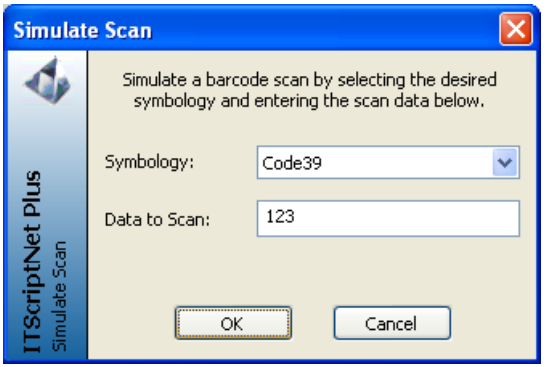
After creating a data collection program, it is a good idea to use the simulator. This will allow you to review the prompts so that you can easily identify any changes you wish to make. It is easier and faster to test your program from the simulator screen than it is to test with the portable terminal.



You can start the simulator by clicking on the *Simulate* menu item found under the *Terminal* main menu item, or by clicking on the simulate icon on the toolbar.

The display of the simulator accurately reproduces the terminal screen. The simulator will display each prompt and will use the prompt's settings to emulate how the program will operate once it is sent to the terminal.

For single prompts, click the **Enter** button to simulate the user pressing the ENTER button on the terminal. The **Escape** button will simulate the terminal's ESCAPE key. For Multi-Prompts (available in the Plus and OMNI packages), the program designer must include buttons or other means for the user to advance from one prompt to the next.



To simulate a scan, press the **Scan** button. This displays the Scan Simulator screen, which allows you to select a barcode symbology and enter the data. This simulates a barcode scan and returns the data along with the symbology to the simulator.

Image capture or text capture prompts will display a message box as a placeholder in the simulator.

The **Download** button will simulate the process of retrieving the data from the terminal by taking the data entered in the simulator and performing the

defined procedure with the data. If the program is configured to create a text file, the simulator will create a text file. If the program is configured to append data into an Access table, the simulator will do the same. The simulator thus reviews and tests not only the data collection process, but will review and test the data processing also. The simulator will allow you to run trial end-to-end tests of your data collection solution.

Prompts Menu

The *Prompts* menu has options to allow you to control how your prompts will be displayed and has access to the *Properties* for each of your prompts in your data collection program. Each of these menu items is described in more detail in the *Single-Prompt* and *Multi-Prompt* sections of this User Guide.

Insert Before / Insert After

These options insert a new prompt before (or after) the currently selected prompt.

Remove

This option removes the currently selected prompt.

Move Up / Move Down

This option changes the sequence of the prompts by swapping the selected prompt with the one above it for *Move Up*, or with the one below it for *Move Down*.

Properties

The *Properties* menu item under the *Prompts* main menu item is another means of accessing the *Prompt Settings* screen.

View Menu

The items in the view menu control aspects of the **IT*Script*Net** design environment.

Advanced

When the *Advanced* menu option is enabled (with a checkmark) the main application window will display the selected prompt's settings and advanced settings. If the Advanced feature is not checked (disabled), the advanced prompt settings will not be displayed.

Side Banners

This option toggles the descriptive side banners along the left edge of each screen. These banners display the screen name as well as the software edition. You can turn the banners on or off with this option.

Toolbar

This menu option will toggle the toolbar on and off. The default is to show the toolbar.

Elements

This option toggles the Element toolbar along the right edge of the screen. Elements are objects such as Text Input fields, Comboboxes, Images, etc. that can be added to Multi-Prompts. Multi-Prompts and Elements apply to **IT*Script*Net Plus** and **IT*Script*Net OMNI** only.

Status Bar

This menu option will toggle the status bar on and off. The default is to show the status bar.

Script Tree

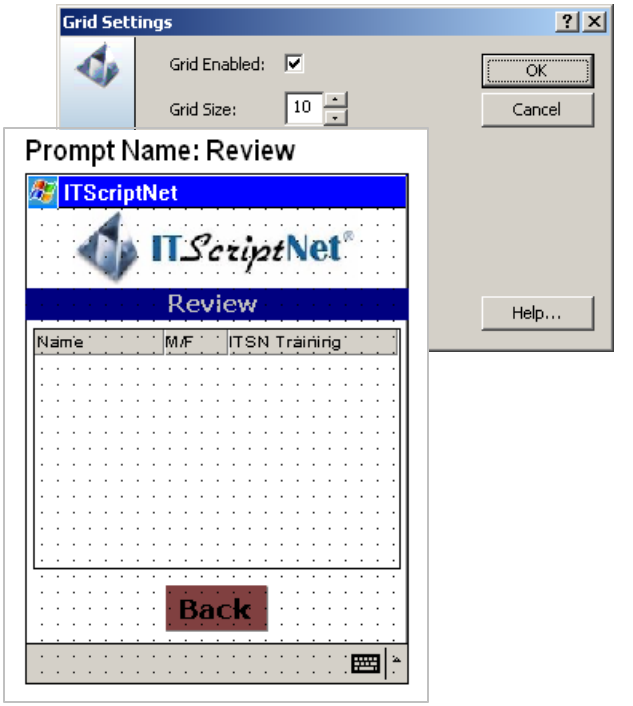
This option toggles the Script Tree view on the main screen. The script tree shows the prompts, elements, and In-Prompt scripts used in the data collection program. It is a compact and very useful view of the data collection program. More information about the Script Tree can be found in both the *Single-Prompt* and *Multi-Prompt* sections of this User Guide.

Snap To Grid

This option toggles whether multi-prompt elements are positioned according to a grid, or freeform. With *Snap To Grid* on, the elements are placed on the grid. If *Snap To Grid* is off, elements can be placed freeform in the main design area.

Grid Settings

This screen allows you to control the Snap-To-Grid settings. You can enable or disable the snap feature, and set the grid size (in pixels). The grid is an optional tool to assist with designing using Multi-Prompt elements. With the grid on, the main design area will show a grid that you can use to help you position your elements.



Language Support

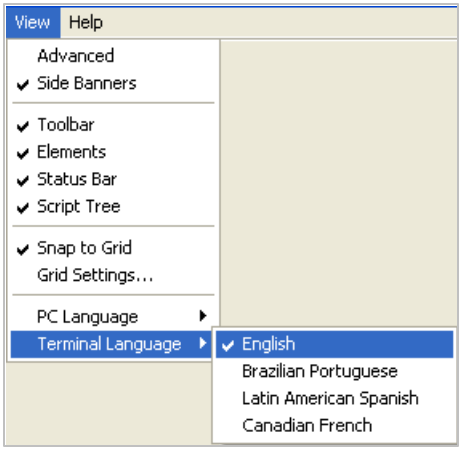
IT*Script*Net supports multiple languages on the handheld terminals. This allows you to change the language of the menus and error messages that your terminal operators will see. At this time, all PC programs are in English.

Changing the Terminal Language

Each PC program has a menu selection to change the terminal language. The language selection is saved so each time you open the program the language setting is retained.

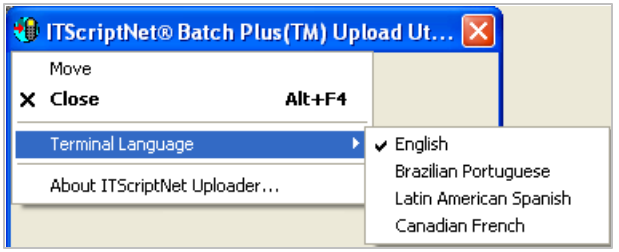
Generator

The **IT*Script*Net** generator has a language selection option on the View menu.



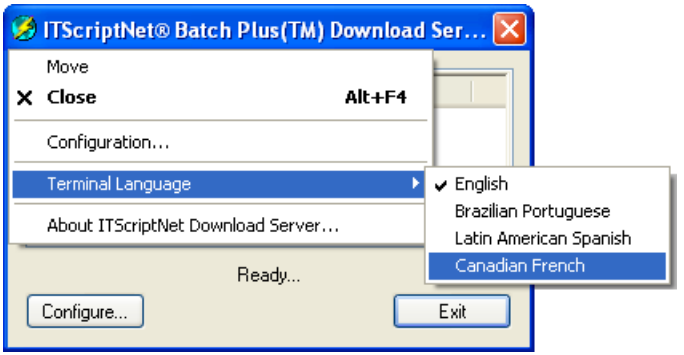
Upload Utility

The **IT*Script*Net** Upload Utility has a language selection option on the System Menu.



Autodownload Utility

The Autodownload utility has a language selection option on the System Menu.



Terminal Support for Languages

When you upload a program to the terminal from the PC, the language file for the selected language will be sent to the terminal. You may need to reboot the terminal or exit and restart the **IT*Script*Net** client for the language changes to take affect.

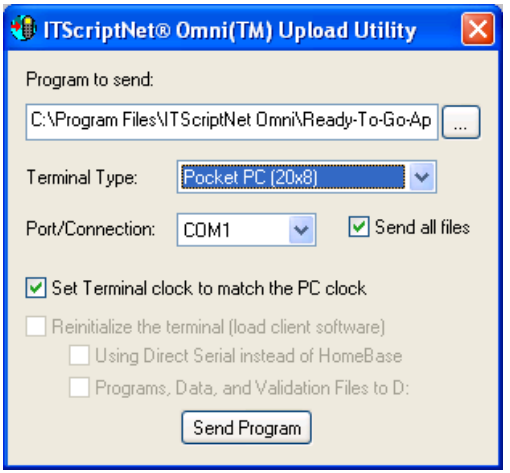
All menus, error messages, and configuration screens will be displayed in the selected language. The prompts in your **IT*Script*Net** programs will not be changed, so you will need to design your programs in the language you want the terminal operator to see.

The Language files are installed into the terminal-specific subdirectories under the main **IT*Script*Net** program directory. The terminal language files have a file extension that corresponds to the ISO 3 letter code for the language. When an ITB is sent to the terminal, the correct language file is renamed to a BIN file and sent to the terminal.

The language for the terminal is determined only by the terminal language selection described in this section. The design of the data collection program (ITB file) does not determine the terminal language.

Upload Utility

The *Upload Utility* can be used to load a program in the terminal.



Program to Send

Select the program to send using the '**...**' button or by typing the file name in the box.

Terminal Type

Select the correct Terminal Type to match the portable data collection terminal you are using.

Port/Connection

You must specify which port or connection to use to connect to your terminal from the options available to you. The options available will depend on your terminal type, the available ports on your PC, and which edition of **IT*Script*Net** that you have. The available ports or connections will be displayed in the drop-down list. The COM and USB ports are physical ports on the PC. The Microsoft ActiveSync Connection will use an existing ActiveSync link to a terminal. ActiveSync itself has options that specify the port within ActiveSync. The RF connection option will use a wireless LAN connection to the terminal and therefore is only an option available in the **IT*Script*Net OMNI** package. Regardless of the port/connection selected, the terminal must be configured to match. Refer to the terminal section for your terminal for more details on configuring the terminal for communication. Note that for serial communications, the baud rate and communication settings are not configurable – they will be set to the factory default settings that come set in the terminal.

Send All Files

By default, the Send All Files option will be checked so that the upload will send the data collection program, its validation files, and all supporting files to the terminal. However, not all support files always need to be sent. The *Support Files* screen in the Generator application allows the program designer to designate which support files should always be uploaded with the data collection program.

Set Terminal Clock

There is an option to send the date and time to the client terminal. The clock option defaults to 'on' so that the client terminal's clock is kept up-to-date. This is important for data collection since the date and time of each data collection record is recorded. This option can be unchecked to not send the date and time, if desired.

Reinitialize (Load Terminal Client)

When sending your program to an **IT*Script*Net** supported DOS terminal, you can enable the option to reinitialize the terminal in addition to sending your data collection program. Reinitializing the terminal will send the **IT*Script*Net** Client software to the terminal in addition to your data collection program. **IT*Script*Net** Client software is a program that runs on the terminal and works with your data collection program to prompt the user, collect responses, and collect and store the data. You must enable the 'Reinitialize Terminal' option the first time that you send your program to the terminal in order to load the **IT*Script*Net** Client Software. After the first time, it will generally not be necessary to reinitialize the terminal unless you need to install an upgrade for the **IT*Script*Net** Client Software. Only the DOS terminals have the Reinitialize option. The Windows CE-based devices and the Pocket PC devices require that you load the client software through ActiveSync, so it is not an option when sending data to the terminal. Please see the section in this User Guide for your type of terminal for more specific information.

Send Program

Click on the **Send Program** button to begin the process of sending your program. You will need to follow the directions on your screen, as the upload procedure is specific to each terminal type. You can create a shortcut to this utility and specify the name of the ITB file as the command line parameter. This will automatically fill the *Program To Send* field with the name of the ITB file you want to send.

Command Line Parameters

There is a command line parameter that you can use to generate validation files and index files without uploading the data to the portable terminal. The command line takes the format

ITBatchULxxxx /Vy <filename.itb>

Where xxxx indicates the edition of the software, and Vy is one of the following:

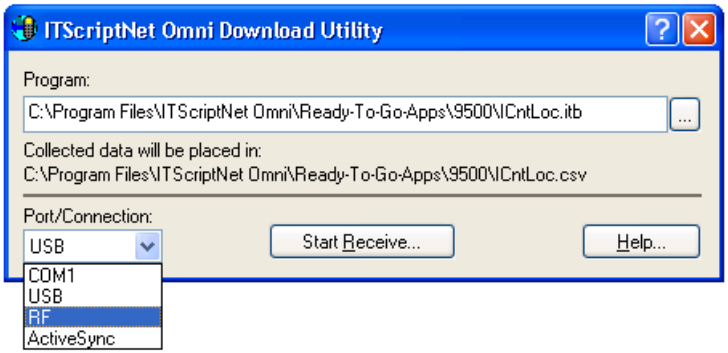
/VU	Generate validation files marked as 'On Every Upload'
/VM	Generate validation files marked as 'Manual'
/VA	Generate validation files marked as 'On Every Upload' or 'Manual'
/V	Same as /VA

After each file is generated, index files will also be created, as specified by the Index File Autogeneration setting. After the files are generated, the Upload Utility exits.

Download Utility

The Receive Data (Download) Utility may be used to retrieve data from the terminal.

Program



You must specify the name of the program whose data you wish to retrieve. Note that the receive screen will display the location where it will place the collected data file. You can also create a shortcut to this utility and specify the name of the ITB file as the command line parameter. This will automatically fill the *Program* field with the name of the ITB file you want to retrieve. Creating a shortcut such that the user does not have to browse for the data collection program will save a user some time when downloading data.

Port/Connection

You must also specify which port or connection to use to connect to your terminal from the options available to you. The available ports or connections will be displayed in the drop-down list. The options available will depend on your terminal type and the available ports on your PC. The COM and USB ports are physical ports on the PC whereas the Microsoft ActiveSync Connection will use an existing ActiveSync link to a terminal. ActiveSync itself has options that specify the port within ActiveSync. The RF connection option will use a wireless LAN connection to the terminal and therefore is only an option available in the **IT*Script*Net OMNI** package. Regardless of the port/connection selected, the terminal must be configured to match. Refer to the terminal section for your terminal for more details on configuring the terminal for communication.

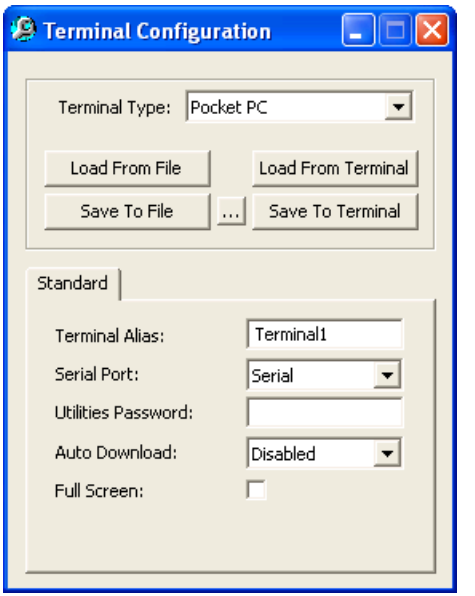
Start Receive

When you are ready, click the **Start Receive…** button to begin the process of retrieving your data. You will need to follow the directions on your screen, as the download procedure is unique to each kind of terminal. For more help, please see the section in this User Guide for your specific type of terminal.

Terminal Configuration Utility

The Terminal Configuration utility is used to manage the configuration of terminals from the PC.

Terminal



Terminal Type

Select the type of terminal to configure. This determines the communications methods available and the default filename of the configuration file.

Load From File

This button allows you to browse for a copy of the terminal configuration file on the PC and load it. This can be used to store a standard configuration that is applied to multiple terminals.

Save To File

This button allows you to save the current configuration to a disk file on the PC. This can be used to store a standard configuration to be applied to multiple terminals.

Load From Terminal

This option attempts to load the configuration directly from the terminal. If the terminal is a Windows CE or PocketPC device, you must establish an ActiveSync connection to the terminal first, and then this utility will use RAPI to connect directly to the terminal and load the configuration. If the terminal is DOS based, this option uses YModem to transfer the configuration file from the terminal.

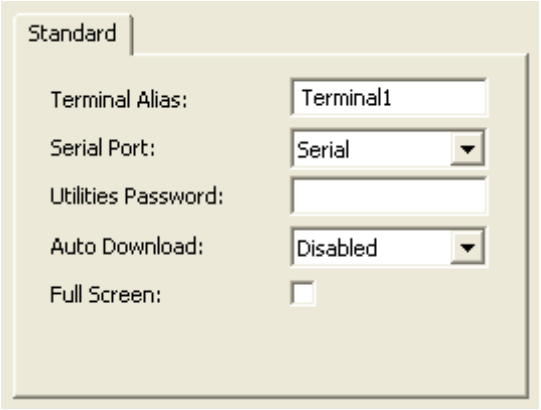
Save To Terminal

This option attempts to save the configuration directly to the terminal. If the terminal is a Windows CE or PocketPC device, you must establish an ActiveSync connection to the terminal first, and then this utility will use RAPI to connect directly to the terminal and save the configuration. If the terminal is DOS based, this option uses YModem to transfer the configuration file to the terminal. Note: Changes will not take effect on the terminal until the client software is restarted.

[...]

This option sets the configuration filename on the terminal. You would not normally need to change this, as the utility sets it automatically when you select the terminal type. However, if you installed the client into a different directory than the default, you can use this option to change the path to the configuration file.

Standard Tab



Terminal Alias

This is the alias for the terminal that is saved in the collected data records. If you leave this field blank, the terminal will construct an alias. See the terminal section of this User Guide for details on this setting.

Serial Port

This option allows you to select the communications port that the terminal will use for upload and download. The options available here will depend on the terminal type selected.

Utilities Password

This option sets a password that the terminal operator must enter to access the utilities menu or exit the client software.

Full Screen

On Windows CE and PocketPC devices, this option sets the client into full screen mode. For more information on this setting, see the terminal specific section of this User Guide.

Using Auto-Download

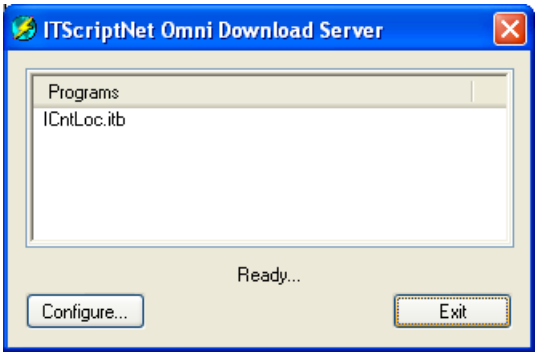
IT*Script*Net supports an auto-download feature that simplifies retrieving data from the portable terminal for programs in manual data collection mode. Using Auto-Download, the operator simply places the portable terminal in the Homebase, and the data will be retrieved automatically. The user does not have to select any menu items or click any buttons. All of the data processing that would normally be performed by the download utility will still be done. You can also configure the auto-download utility to resend the program and any validation files and/or support files back to the terminal after the data has been retrieved. This allows you to keep your data files current on the terminal without having to manually upload them.

You can also configure Auto-download so that the data does not download immediately when the terminal is placed in the cradle, but when the operator selects Send Data to PC from the terminal Main Menu. The operator does not need to do anything at the PC. See the section on Manual Downloading for more details.

In order for a data collection program to auto-download, the **Download Server** (from the **IT*Script*Net** Programs group on the PC) must be configured to perform auto-download for that data collection program, and the terminal must be configured to select the auto-download port. When the portable terminal is placed in the Homebase, the terminal and the Download Server exchange information about which programs have collected data and which programs are allowed to auto-download. This allows you to permit certain programs to use this feature while excluding others.

Normally, after downloading data, the terminal will prompt the user to ask whether the collected data can be deleted. During autodownload, however, the collected data will normally be deleted from the terminal automatically without a prompt to the user. This default behavior is intended to minimize the amount of user intervention required. If it is desirable to have the user be prompted prior to the data being deleted from the terminal after the autodownload, there is an option on the *Configure Receive* screen in the Generator Application that can turn on the prompt before deleting.

The Download Server runs as a Task Tray icon, so it can run in the background on your PC. However, when the data is being downloaded, the download status screen is displayed. When the program is running in the Task Tray, you can double-click its icon to bring up the main screen again. Clicking the X button on the screen minimizes the Download Server to the Task Tray but does not exit. To completely exit the Download Server, you must click the **Exit** button on the Download Server main screen. Only one instance of the download server should be running at a time.



The main screen for the Download Server is shown. Any program that has been configured for Auto-Download will be listed.

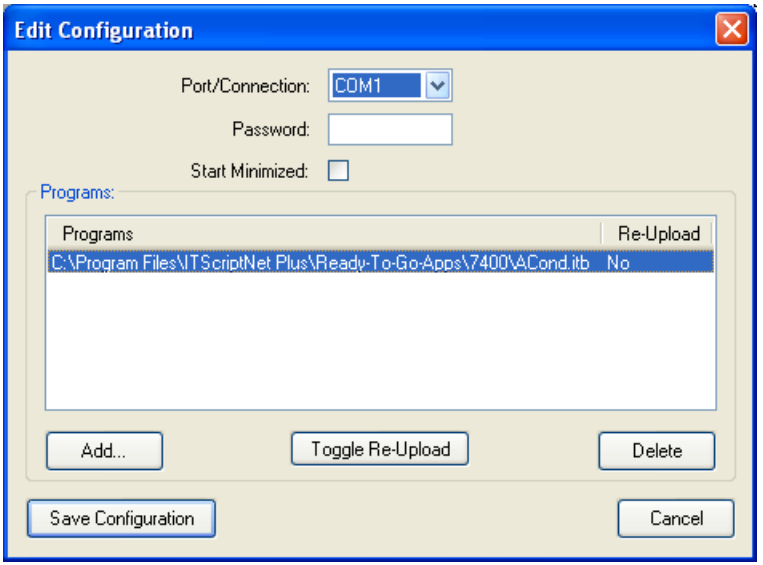
Exit

Click Exit to completely shutdown the Download Server. No data will be downloaded once you exit the Download Server.

Configure Auto-Download

This button brings up the *Edit Configuration* screen. This screen is used to determine which programs can use auto-download. Use the **Add** and **Delete** buttons to add or remove programs from the auto-download list.

Each program in the auto-download list is configured separately to Re-Upload. Select a program from the list and click the **Toggle Re-Upload** button to change the Re-Upload state from No to Yes. If Re-Upload is set to Yes, the program and its associated files (validation files, index files, support files, etc.) will be resent to the terminal after the data is downloaded and processed. The Re-Upload feature can be used to keep the data collection program and the validation files up-to-date.



The Port/Connection drop-down is used to select the serial port or USB port from your PC to your terminal's docking cradle. The port you select must not be in use by any other program or the Download Server will report an error. Auto-download can use Serial ports, Direct USB, or ActiveSync.

The Password field allows you to specify a password that must be entered by anyone who clicks the **Configuration** button or attempts to Exit the Download Server. If the password field is left blank, no password prompt will be displayed.

Click the **Save Configuration** button to save your changes, or click **Cancel** to exit without saving.

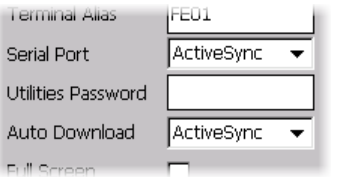
Configure the Terminal

The terminal must be configured to allow Auto-downloading and must match the port/connection specified in the Download Server. Please refer to the terminal-specific user guide for the terminal

type you are using. You will need to use the client configuration screen to select the port to use for Auto-download.

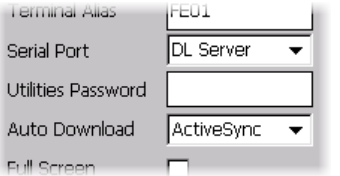
There are two communications port selections on the client configuration screen. The Serial Port is used to select the communications method used for normal upload / download with the Upload or Download utilities. The Auto Download port is used to select the communications method to use with the Auto Download Server. The exception is the DL Server selection for Serial Port, which is used to control automatic or manual downloads.

Automatic Downloading



To enable automatic downloading, the Auto Download serial port must be set to the communications method you are using. The Serial Port can be any setting except DL Server. In this case, the terminal will automatically try to connect to the Auto Download server as soon as it is placed in the cradle, using the port specified in the Auto Download setting. You will still be able to communicate manually with the upload and download utilities.

Manual Downloading



To enable manual downloading using the Auto-Download server, the Auto Download serial port must be set to the communications method you are using. The Serial Port must be set to DL Server. When you place the terminal in the cradle, it will not automatically try to connect to the Auto Download server. You can use the LoadProgram or SendData to PC buttons to manually start the communications. You can also use the *Load Program* or *DownloadData* script functions to initiate communications. With this configuration, you can not use the standard upload / download utilities to communicate with the terminal.

Using the ActiveX Controls

IT*Script*Net supports two ActiveX controls that can be used to Upload a data collection program and its associated files to a portable terminal, or to download collected data from the terminal. These ActiveX Controls can be used by programmers to embed the **IT*Script*Net** upload and download functionality into Visual Basic, Visual C/C++, or other development environments in order to seamlessly integrate **IT*Script*Net** as part of a full solution.

Download ActiveX Control -- ITBatchDLX

StartDownload Method

The StartDownload method is used to retrieve collected data from the terminal.

*long **StartDownload**(long Port, LPCTSTR Program, long Quiet)*

Port

The first parameter to the StartDownload method is the COM port to use. See the table for valid port IDs.

Value	Port
1, 2, 3, etc	COM1, COM2, COM3, etc
-1	Ir Sockets
-2	RF
-3	ActiveSync

Program

The second parameter to the StartDownload method is the full path to the ITB file whose data you want to retrieve and process.

Quiet

A flag indicating whether to display or suppress the 'Download Complete' message is the third parameter to the StartDownload method. If the Quiet flag is 0, the message will be displayed after download. If the Quiet flag is 1, the message will not be displayed. Note: Even if Quiet is selected, error messages will be displayed.

Return Value

The StartDownload returns a long integer 1 if the download is successful, 0 if not.

Upload ActiveX Control

Send Program Method

The Send Program method is used to send a data collection program and its associated files (validation files, index files, etc.) to the terminal.

long **SendProgram**(long Port, LPCTSTR Program, long TermType,
long SendClock, long SendClient, long UseSerial)

Port

The first parameter to the SendProgram method is the COM port to use. See the table for valid port IDs.

Value	Port
1, 2, 3, etc	COM1, COM2, COM3, etc
-1	Ir Sockets
-2	RF
-3	ActiveSync

Program

The second parameter to the SendProgram method is the full path of the ITB file to send to the terminal.

TerminalType

The TerminalType parameter specifies the type of terminal to which you are sending the program. To find the terminal type for your specific terminal model, see the terminal-specific documentation located in the **Documentation -> Client Guides** program group.

SendClock

The SendClock parameter specifies whether to set the terminal clock to match the PC clock. A non-zero value will set the clock; a value of 0 will not set the clock.

SendClient

The SendClient parameter specifies whether to send the client to the terminal. Note: This only applies to DOS-Based clients. This parameter is ignored if set for Windows CE or PocketPC based terminals. A non-zero value will send the client; a value of 0 will not send the client.

UseSerial

The UseSerial parameter specifies whether to use the serial port instead of HomeBase for DOS-Based terminals. This parameter is ignored if set for any other terminal. Set to non-zero for serial cable uploads or 0 to use a Homebase.

Return Value

The SendProgram returns a long integer 1 if the transfer is successful, or 0 if it fails.

PC Client ActiveX Control

The PC Client ActiveX control can be used to embed data collection in your PC application. This control is the same one used by the PC Client Application that is a part of the ITScriptNet programs. The PC Client ActiveX control can be used to run the same data collection programs that you use for your portable terminals, or programs designed specifically for use on a PC.

SetITBFileMethod

The SetITBFile method is used to specify the data collection program to execute.

long **SetITBFile**(BSTR ITBFile)

ITBFile

The fully-qualified pathname of the ITB file.

Return Value

If the program loads successfully, the return value is non-zero. If there is an error loading the program, the return value is 0.

CollectData

The CollectData method is used to execute a data collection program.

long **CollectData**( )

Return Value

The return value is always 0.

SetSkin

The SetSkin method is used to specify the look of the data collection program by emulating a specific terminal type.

long **SetSkin**(long SkinID)

SkinID

The ID of the skin to use. The Skin ID matches the terminal type specified in the terminal-specific manuals. Use 0 for the PC Client skin.

Return Value

The return value is always 0.

SetAlias

The SetAlias method is used to specify the Terminal Alias to use when collecting data.

long **SetAlias**(BSTR Alias)

Alias

The alias to use for data collection.

Return Value

The return value is always 0.

ProcessCollectedData

The ProcessCollectedData method is used to process collected data as though it had been downloaded from a portable terminal.

long **ProcessCollectedData**( )

Return Value

The return value is always 0.

SetServerParams

The SetServerParams method is used to specify a connection to an Omni Server. This method is available in **IT*Script*Net OMNI** only.

long **SetServerParams**(LPCTSTR ServerAddress, long ServerPort, long ServerTimeout)

ServerAddress

The IP Address of the OMNI Server, in the form “xxx.xxx.xxx.xxx”.

ServerPort

The IP Port of the OMNI Server. The default is 61200.

ServerTimeout

The timeout value to use when connecting to the OMNI server. This value is in milliseconds.

Return Value

The return value is always 0.

GetITBHeight

The GetITBHeight method is used to retrieve the height that the ITB file is designed for. If the program was designed for a portable data collection terminal, this will be the height of the device screen. If the terminal type allows for a configurable screen size, this will be the height specified in the program design.

long **GetITBHeight**( )

Return Value

The return value is the height of the ITB design.

GetITBWidth

The GetITBWidth method is used to retrieve the width that the ITB file is designed for. If the program was designed for a portable data collection terminal, this will be the width of the device screen. If the terminal type allows for a configurable screen size, this will be the width specified in the program design.

long **GetITBWidth**( )

Return Value

The return value is the width of the ITB design.

SetCollectFile

The SetCollectFile method is used to set the filename used to store the collected data. By default, te collected data file will have the same name as the ITB file, with an ITC extension.

long **SetCollectFile**(BSTR CollectFile)

CollectFile

The filename to use for the collected data.

Return Value

The return value is always 0.

EnableScanning

The EnableScanning method is used to turn on barcode scanning using a hand scanner connected to the PC serial port. The parameters of this method set the serial port, baud rate, data bits, and parity.

long **EnableScanning**(long lScannerType, long lPort, long lBaud, long lBits, char cParity)

lScannerType

The type of scanner to use. 0 indicates a generic serial scanner, and 1 indicates a HandHeld Products 3800.

lPort

The serial port ID that the scanner uses (1 for COM1, etc).

lBaud

The baud rate to use for the scanner.

lBits

The number of data bits to use for the scanner.

cParity

The parity to use for the scanner.

Return Value

The return value is always 0.

DisableScanning

The DisableScanning method is used to turn off barcode scanning.

long **DisableScanning**( )

Return Value

The return value is always 0.

Sample Code

This sample code demonstrates using the PC Client ActiveX control in VB.Net

```
AxITClientX1.SetAlias("PCClientX")
AxITClientX1.SetSkin(0)
AxITClientX1.SetITBFile("C:\Temp\PCShell.itb")
AxITClientX1.SetServerParams("192.168.3.1", 61200, 5000)
AxITClientX1.EnableScanning(1, 1, 9600, 8, Asc("N"))
AxITClientX1.CollectData()
```

Error! Not a valid filename.

String Functions

Exact

Syntax:	Exact(<String1> , <String2>)
Parameters:	
<String1>	The first string to test
<String2>	The second string to test
Returns:	Returns 1 if <string1> exactly matches <string2> (except for case), returns 0 otherwise.
Examples:	Exact(@MyString@,"Hello") = 1 if @MyString@ is a variable with the value of "Hello"
Notes:	The string comparison is case insensitive.

InStr

Syntax:	InStr(<Start>, <String> , <SearchFor>, <CaseSensitive>)
Parameters:	
<Start>	Starting position to search <String>. The first character is position 1.
<String>	The string to search
<SearchFor>	The string to search for within <String>
<CaseSensitive>	1 to perform a case-sensitive search; 0 ignores case
Returns:	Returns the position of the first match from the start position of the first character in the SearchFor string. Returns 0 if SearchFor string is not found.
Examples:	InStr(1,"Hello World","o",1) returns 5.
Notes:	Locates the string <SearchFor> within <String>. If found, the position of the first character of the matching string is returned.

InStrRev

Syntax:	InStrRev(<Start>, <String> , <SearchFor>, <CaseSensitive>)
Parameters:	
<Start>	The character position to start searching from
<String>	The string to search
<SearchFor>	The string to search for within <String>
<CaseSensitive>	Whether to perform a case-sensitive search
Returns:	The character position of the first character of the matching string in <String>.
Examples:	InStrRev(1,"Hello World","o",1) returns 8
Notes:	Same as InStr, but the search works from right to left. The search always starts from the right end of the string, but stops searching at <Start>.

IsNumeric

Syntax:	IsNumeric(<Expression>)
Parameters:	
<Expression>	A string containing the characters to be tested.
Returns:	Returns 1 (same as TRUE) if the expression is numeric (contains no alpha characters), else 0 (same as FALSE).
Examples:	IsNumeric("7") returns TRUE
Notes:	The digits 0 - 9 and '.' are considered numeric. Leading plus or minus signs are allowed. Everything else causes a FALSE return.

LCase

Syntax:	LCase(<String>)
Parameters:	
<String>	The string to be converted to lowercase
Returns:	Returns <String> converted to all lowercase
Examples:	LCase("Hello") returns "hello".
Notes:	All alphabetic characters are converted to lowercase. Numeric and punctuation characters are not changed.

Left

Syntax:	Left(<String> , <Num>)
Parameters:	
<String>	The string from which the characters should be returned
<Num>	The number of characters to be returned
Returns:	Returns a substring of <String> containing the left <Num> number of characters.
Examples:	Left("Hello",2) returns "He"
Notes:	The <String> parameter may be a string or numeric expression.

Len

Syntax:	Len(<String>)
Parameters:	
<String>	The string whose length is to be calculated.
Returns:	Returns the number of characters in <String>.
Examples:	Len("Hello") returns 5.
Notes:	The <String> parameter may be a string or numeric expression.

LTrim

Syntax:	LTrim(<String>)
Parameters:	
<String>	The string to trim
Returns:	Returns <String> with all leading spaces removed.
Examples:	LTrim(" Hello ") returns "Hello ".
Notes:	The <String> parameter may be a string or numeric expression.

Mid

Syntax:	Mid(<String> , <Start>, <Num>)
Parameters:	
<String>	The string from which the characters should be returned.
<Start>	The starting position within the string. The first character is position 1.
<Num>	The number of characters to return
Returns:	Returns a substring of <String> starting with the <Start> position with a length of <Num> characters.
Examples:	Mid("Hello", 2, 3) returns "ell".
Notes:	The <String> parameter may be a string or numeric expression.

Pad

Syntax:	Pad(<String>, <Length>)
Parameters:	
<String>	The string to be padded
<Length>	The number of characters of the returned string
Returns:	The <String> padded with spaces to the length <Length>
Examples:	Pad("Hello",10) returns "Hello ".
Notes:	Adds trailing spaces to force the string to be the length specified.

Replace

Syntax:	Replace(<String>, <Replace>, <ReplaceWith>, <CaseSensitive>)
Parameters:	
<String>	The string in which replacements are to be made
<Replace>	The string to replace
<ReplaceWith>	The string to replace with
<CaseSensitive>	Whether to perform a case sensitive search
Returns:	Returns the string with <Replace> replaced with <ReplaceWith>
Examples:	Replace("ABC 123 xyz", "123", "9", 0) returns "ABC 9 xyz"
Notes:	Set <CaseSensitive> to 1 to enable case-sensitive replacement, else set it to 0 to allow replacement regardless of case. This function replaces all instances of the <Replace> string.

Rept

Syntax:	Rept(<String>, <Number>)
Parameters:	
<String>	The string to repeat
<Number>	The number of times to repeat the string
Returns:	Returns a string that contains <String> repeated <Number> times
Examples:	Rept("Hello",3) returns "HelloHelloHello"
Notes:	The parameter <String> may be string or numeric data. The <Number> parameter will be treated as an integer.

Right

Syntax:	Right(<String> , <Num>)
Parameters:	
<String>	The string from which the characters should be returned
<Num>	The number of characters to be returned
Returns:	Returns a substring of <String> containing the right <Num> of characters
Examples:	Right("Hello",2) returns "lo"
Notes:	The <String> parameter may be a string or numeric expression.

RTrim

Syntax:	RTrim(<String>)
Parameters:	
<String>	The string to be trimmed
Returns:	Returns <String> with any trailing spaces removed
Examples:	RTrim(" Hello ") returns " Hello"
Notes:	Trims trailing spaces

Search

Syntax:	Search(<String> , <Search> , <StartPos>)
Parameters:	
<String>	The string to search
<Search>	The characters to search for
<StartPos>	The starting position within <String> to search.
Returns:	Returns the position that any character from <Search> appears within <String>, starting from <StartPos>. If no character from <Search> is found in <String>, the length of the string is returned.
Examples:	Search("Hello World", "eo", 1) returns 2.
Notes:	Searches for any one of a set of characters within a string.

Space

Syntax:	Space(<Length>)
Parameters:	
<Length>	The number of spaces to format into the string
Returns:	Returns a string containing spaces of the length specified
Examples:	Space(5) returns " "
Notes:	Creates a string of <Length> spaces

Split

Syntax:	Split(<String> , <Char>, <Result1>, <Result2>, ...)
Parameters:	
<String>	The string to split
<Char>	The character at which to split.
<Result1>	The string to receive the first substring
<Result2>	The string to receive the second substring
	You may specify any number of result strings to receive the data.
Returns:	Returns the number of strings parsed. The data will be in Result1, Result2, etc.
Examples:	Split("Hello World"," ",Result1, Result2) returns 2 and puts "Hello" in Result 1 and "World" in Result2.
Notes:	Parse the <String> at the character specified by <Char> into the result strings <Result1>, <Result2>, etc. Any number of result strings may be specified.

SplitN

Syntax:	SplitN(<String> , <Char>, <Index>)
Parameters:	
<String>	The string to split
<Char>	The character at which to split.
<Index>	Specifies which substring to return.
Returns:	Returns the value in the <Index> position.
Examples:	SplitN("ABC!DEF!GHI","!", 2) returns "DEF".
Notes:	Parse <String> using the <Char> as the separator, returning the value in the <Index> position. The first value is referenced by <Index> 1.

StrComp

Syntax:	StrComp(<String1> , <String2>)
Parameters:	
<String1>	The first string to compare
<String2>	The second string to compare
Returns:	Returns 1 if <String1> is greater (sorts after) <String2>. Returns 0 if <String1> and <String2> match. Returns -1 if <String1> is less than (sorts before) <String2>.
Examples:	StrComp("Hello","World") returns -1
Notes:	This comparison is case-sensitive.

StrDup

Syntax:	StrDup(<Length> , <String>)
Parameters:	
<Length>	The number of times to repeat <String>
<String>	The string to be repeated
Returns:	Returns a string that contains the first character of <String> repeated <Length> times.
Examples:	StrDup(5, "Hello") returns "HHHHH"
Notes:	The <String> parameter may be a string or numeric expression.

Subst

Syntax:	Subst(<String>, <Start>, <Length>, <Replace>)
Parameters:	
<String>	The string to replace in
<Start>	The starting position of the replacement within <string>
<Length>	The length to replace
<Replace>	The replacement string
Returns:	Returns <String> with the <Length> number of characters from the <Start> position replaced by the string in <Replace>.
Examples:	Subst("Collect Asset Data", 9, 5, "Inventory") returns "Collect Inven Data".
Notes:	Both <String> and <Replace> may be string or numeric expressions.

Text

Syntax:	Text(<Data>, <Mask>)
Parameters:	
<Data>	The source data to be formatted
<Mask>	The mask to use for formatting the text.
Returns:	Returns the data formatted according to the mask
Examples:	Text("4405551212","(???) ???-????") = "(440) 555-1212"
Notes:	The Mask character is ?. Any other character is literal. You may use the escape char: \, i.e. \? = literal ?.

Trim

Syntax:	Trim(<String>)
Parameters:	
<String>	The string to be trimmed
Returns:	The <String> with all leading and trailing spaces removed
Examples:	Trim(" Hello ") returns "Hello"
Notes:	Trims leading and trailing spaces

UCase

Syntax:	UCase(<String>)
Parameters:	
<String>	The string to be converted to uppercase
Returns:	Returns <String> will all alpha characters converted to uppercase
Examples:	UCase("Hello") returns "HELLO".
Notes:	Converts the string to all uppercase

Conversion Functions

Asc

Syntax:	Asc(<Char>)
Parameters:	
<Char>	A string containing the character to be converted
Returns:	Returns the ASCII character code for the first character in <Char>
Examples:	Asc("A") = 65
Notes:	Converts a character to its ASCII equivalent

Chr

Syntax:	Chr(<Num>)
Parameters:	
<Num>	Numeric value to be converted
Returns:	Returns the character associated with a specified ASCII character code
Examples:	Chr(65) = "A"
Notes:	The value of <Num> should be between 0 and 255

Format

Syntax:	Format(<Num> , <Numdecimals>, <Sep>, [thousep], [decsep])
Parameters:	
<Num>	The number to format
<Numdecimals>	How many decimal places to keep
<Sep>	Whether to use a thousands separator
[thousep]	Optional character to use as thousands separator
[decsep]	Optional character to use as decimal separator
Returns:	Returns the number as a string with the number of decimals specified
Examples:	Format(@Num@, 2, 1) = "17,345.21" where @Num@ = 17345.2142
Notes:	Set the <Sep> parameter to 1 to include ',' as the thousands separator. If <Sep> is set to 0, no separator will be used. Optionally, you can specify the Thousands separator and Decimal Separator. The default Thousands separator is ',' and the default Decimal separator is '.' if not specified.

Val

Syntax:	Val(<String>)
Parameters:	
<String>	The string to be converted
Returns:	Returns an integer representation of the string in <String>
Examples:	Val("124a6") returns 124
Notes:	Converts the string from left to right, stopping at the first non-numeric character.

Logical Functions

And

Syntax:	AND(<Expression1> , <Expression2>)
Parameters:	
<Expression1>	Logical expression to be evaluated
<Expression2>	Logical expression to be evaluated
Returns:	Returns 1 (same asTRUE) if <Expression1> and <Expression2> are both non-zero, else 0 (same as FALSE).
Examples:	AND(IsNumeric("5"), IsNumeric("7")) returns 1 (TRUE)
Notes:	AND is a function, not an operator. The parameters should evaluate to logical expressions. Before testing, VAL() will be performed on each expression. Any non-numeric value is considered FALSE. Any non-zero Numeric value is considered TRUE. Values starting with numeric digits will be converted up to the first non-numeric character.

IIF

Syntax:	IIF(<Expression> , <Result1> , <Result2>)
Parameters:	
<Expression>	Logical expression to be evaluated
<Result1>	The expression to be returned if <Expression> is true
<Result2>	The expression to be returned if <Expression> is false
Returns:	Returns <Result1> if <Expression> is true, else returns <Result2>
Examples:	IIF(IsNumeric(@expr@),"Number","Alpha") returns "Number" if @expr@ evaluates to a numeric expression, otherwise "Alpha" is returned.
Notes:	The <Expression> should evaluate to a logical expression. Before testing, VAL() will be performed on <Expression>. Any non-numeric value is considered FALSE. Any non-zero Numeric value is considered TRUE. Values starting with numeric digits will be converted up to the first non-numeric character.

Not

Syntax:	NOT(<Expression>)
Parameters:	
<Expression>	Logical expression to be evaluated.
Returns:	Returns 0 (same as FALSE) if <Expression> is non-zero, else 1 (same as TRUE).
Examples:	NOT(IsNumeric("5")) returns 0 (FALSE)
Notes:	NOT is a function, not an operator. The parameter should evaluate to a logical expression. Before testing, VAL() will be performed on the expression. Any non-numeric value is considered FALSE. Any non-zero Numeric value is considered TRUE. Values starting with numeric digits will be converted up to the first non-numeric character.

Or

Syntax:	OR(<Expression1>, <Expression2>)
Parameters:	
<Expression1>	Logical expression to be evaluated.
<Expression2>	Logical expression to be evaluated.
Returns:	Returns 1 (same as TRUE) if either <Expression1> or <Expression2> are non-zero, else 0 (same as FALSE).
Examples:	OR(IsNumeric("A"), IsNumeric("7")) returns TRUE
Notes:	OR is a function, not an operator. The parameters should evaluate to logical expressions. Before testing, VAL() will be performed on each expression. Any non-numeric value is considered FALSE. Any non-zero Numeric value is considered TRUE. Values starting with numeric digits will be converted up to the first non-numeric character.

Math Functions

Abs

Syntax:	Abs(<Value>)
Parameters:	
<Value>	A numeric value to be converted
Returns:	Returns the absolute value of <Value>, ignoring sign.
Examples:	Abs(3.5) = 3.5 Abs(-3) = 3
Notes:	Conversion stops at the first non-numeric character. Numeric characters are digits, '-', and '.'

Fix

Syntax:	Fix(<Value>)
Parameters:	
<Value>	A number to be converted to an integer
Returns:	The integer portion of <Value>
Examples:	Fix(3.5) = 3, Fix(-3.8) = -3
Notes:	Removes the fractional part of <Value> and returns the resulting integer value. The largest number supported is +/- 2147483648.

Int

Syntax:	Int(<Value>)
Parameters:	
<Value>	A number to be converted to an integer
Returns:	Returns the largest integer which is less than or equal to <Value>
Examples:	Int(3.5) = 3, Int(-3.2) = -4
Notes:	Use Int when you want the number to be less than <Value>. Use Fix to truncate the fractional portion. For positive numbers, Int and Fix return the same result. For negative numbers, they do not. The largest number supported is +/- 2147483648.

Mod

Syntax:	Mod(<Value1> , <Value2>)
Parameters:	
<Value1>	The numerator for the division.
<Value2>	The denominator for the division.
Returns:	Returns the remainder of the division of <Value1> by <Value2>
Examples:	Mod(6,2) = 0 Mod(11,3) = 2
Notes:	The expressions <Value1> and <Value2> will be converted to integer data. The conversion stops at the first non-integer character. The largest number supported is +/- 2147483648.

Quotient

Syntax:	Quotient(<Value1> , <Value2>)
Parameters:	
<Value1>	An integer used for the numerator
<Value2>	An integer used for the denominator
Returns:	Returns the integer results of the division of <Value1> by <Value2>
Examples:	Quotient(6,2) = 3 Quotient(13,3) = 4
Notes:	The expressions <Value1> and <Value2> will be converted to integers. The expressions will be converted up to the first non-numeric character. The largest number supported is +/- 2147483648.

Rand

Syntax:	Rand( )
Parameters:	None
Returns:	Returns a random number between 0 and 32767.
Examples:	Rand( )
Notes:	The random number generator is seeded on the first call to this function.

Round

Syntax:	Round(<Value> , <Numplaces>)
Parameters:	
<Value>	A numeric value to be rounded
<Numplaces>	The number of decimal places to round to
Returns:	Rounds <Value> to the nearest decimal place indicated by <Numplaces>.
Examples:	Round(7257.28456,0) = 7257 Round(7257.28456,2) = 7257.28 Round(7257.28456,-2) = 7300
Notes:	The <Value> will be converted to numeric data. The conversion will stop at the first non-numeric character.

Sgn

Syntax:	Sgn(<Value>)
Parameters:	
<Value>	The numeric data to test for sign
Returns:	-1 if <Value> is less than 0 0 if <Value> is equal to 0 1 if <Value> is greater than 0
Examples:	Sgn(3.5) = 1
Notes:	The <Value> will be converted to numeric data. The conversion will stop at the first non-numeric character.

Sqr

Syntax:	Sqr(<Value>)
Parameters:	
<Value>	A numeric expression
Returns:	Returns the square root of <Value>
Examples:	Sqr(9) = 3 Sqr(8) = 2.828
Notes:	The <Value> will be converted to a numeric value. The conversion stops at the first non-numeric character. If a negative number is supplied, the square root of the absolute value will be taken.

Date/Time Functions

Date

Syntax:	Date([datetime])
Parameters:	
[datetime]	An optional DateTime string. If not specified, the current system date/time is used. Must be in the format MM/DD/YYYY HH:MM:SS.
Examples:	Date() returns "08/18/2005" on August 18th, 2005.
Notes:	Returns the current date if the parameter is not specified.

Day

Syntax:	Day([datetime])
Parameters:	
[datetime]	An optional DateTime string. If not specified, the current system date/time is used. Must be in the format MM/DD/YYYY HH:MM:SS.
Returns:	Returns the day of the month
Examples:	Day() returns 18 on the 18th of the month.
Notes:	Returns the current day if the parameter is not specified.

Hour

Syntax:	Hour([datetime])
Parameters:	
[datetime]	An optional DateTime string. If not specified, the current system date/time is used. Must be in the format MM/DD/YYYY HH:MM:SS.
Examples:	Hour returns 14 from 2:00pm to 2:59pm.
Notes:	The result is always in 24-hour time. Returns the current hour if the parameter is not specified.

Minute

Syntax:	Minute([datetime])
Parameters:	
[datetime]	An optional DateTime string. If not specified, the current system date/time is used. Must be in the format MM/DD/YYYY HH:MM:SS.
Examples:	Minute returns 15 at quarter after each hour.
Notes:	Returns the current minute if the parameter is not specified.

Month

Syntax:	Month([datetime])
Parameters:	
[datetime]	An optional DateTime string. If not specified, the current system date/time is used. Must be in the format MM/DD/YYYY HH:MM:SS.
Examples:	Month() returns 8 in August.
Notes:	Returns the current month if the parameter is not specified.

Now

Syntax:	Now()
Parameters:	None
Returns:	The current date and time, in the format MM/DD/YYYY HH:MM:SS
Examples:	Returns “08/18/2005 23:13:26” at 11:13:26pm on August 18th, 2005.
Notes:	Returns the current time and date.

Second

Syntax:	Second([datetime])
Parameters:	
[datetime]	An optional DateTime string. If not specified, the current system date/time is used. Must be in the format MM/DD/YYYY HH:MM:SS.
Examples:	Second() returns 30 at 30 seconds past each minute.
Notes:	Returns the current second if the parameter is not specified.

Time

Syntax:	Time([datetime])
Parameters:	
[datetime]	An optional DateTime string. If not specified, the current system date/time is used. Must be in the format MM/DD/YYYY HH:MM:SS.
Examples:	Time() returns "23:13:26" at 11:13:26pm.
Notes:	Returns the current time if the parameter is not specified.

Year

Syntax:	Year([datetime])
Parameters:	
[datetime]	An optional DateTime string. If not specified, the current system date/time is used. Must be in the format MM/DD/YYYY HH:MM:SS.
Examples:	Year() returns 2005 in 2005.
Notes:	Returns the current year if the parameter is not specified.

DateDiff

Syntax:	DateDiff(<Datetime1>, <Datetime2>, <Interval>)
Parameters:	
<Datetime1>	Starting date / time
<Datetime2>	Ending date / time
<Interval>	Type of interval to calculate. Valid options are "Y" (years), "M" (months), "D" (days), "H" (hours), "N" (minutes), "S" (seconds).
Returns:	The time interval between the two dates.
Examples:	DateDiff("01/05/2005 11:13:52", "01/05/2005 12:15:33", "H") returns 1.
Notes:	Calculates the time interval from <Datetime1> to <Datetime2>. Note that <Datetime> must be in the format MM/DD/YYYY HH:NN:SS. To get a positive number for the result, <Datetime1> should be less than <Datetime2>.

DateAdd

Syntax:	DateAdd(<Datetime>, <Interval>, <Amount>)
Parameters:	
<Datetime>	Starting date / time.
<Interval>	The type of interval to add. Valid options are "Y" (years), "M" (months), "D" (days), "H" (hours), "N" (minutes), "S" (seconds).
<Amount>	The number to add to the starting date / time.
Returns:	The resulting date time.
Examples:	DateAdd("01/05/2005 11:13:52", "D", 4) returns "01/09/2005 11:13:52".
Notes:	Calculates the date / time resulting from adding the <Interval> to the <Datetime>. Note that <Datetime> must be in the format MM/DD/YYYY HH:NN:SS.

BuildDate

Syntax:	BuildDate(<Year>, <Month>, <Day>, <Hour>, <Minute>, <Second>)
Parameters:	
<Year>	The year to use for building the date / time.
<Month>	The month to use for building the date / time.
<Day>	The day to use for building the date / time.
<Hour>	The hour to use for building the date / time.
<Minute>	The minute to use for building the date / time.
<Second>	The second to use for building the date / time.
Returns:	The resulting date time.
Examples:	BuildDate (2005, 1, 9, 11, 13, 52) returns "01/09/2005 11:13:52".
Notes:	Calculates the date / time resulting from the parameters specified.

DayOfYear

Syntax:	DayOfYear([Datetime])
Parameters:	
[Datetime]	Optional. If specified, this is the datetime to use for calculating the Date of the Year.
Returns:	The day of the year (Julian date) for the specified datetime or current date.
Examples:	DayOfYear ("01/09/2005 11:13:52") returns 9.
Notes:	Returns the Day of the Year, as a number, for a date/time with Jan 1st = 1. If the [Datetime] parameter is specified, it will be parsed for the date. If it is not specified, the current date will be used. Note that [Datetime] must be in the format MM/DD/YYYY HH:NN:SS.

DayOfWeek

Syntax:	DayOfWeek([Datetime])
Parameters:	
[Datetime]	Optional. If specified, this is the datetime to use for calculating the Date of the Week.
Returns:	The day of the week for the specified datetime or current date.
Examples:	DayOfWeek ("01/09/2005 11:13:52") returns 1.
Notes:	Returns the Day of the Week, as a number, for a date/time with Sunday = 1, Monday = 2, etc. If the [Datetime] parameter is specified, it will be parsed for the date. If it is not specified, the current date will be used. Note that [Datetime] must be in the format MM/DD/YYYY HH:NN:SS.

DateCompare

Syntax:	DateCompare(<Datetime1>, <Datetime2>)
Parameters:	
<Datetime1>	The starting date for the comparison.
<Datetime2>	The ending date for the comparison.
Returns:	A number indicating the result of the comparison.
Examples:	DateCompare("01/10/2005 11:13:52", "01/09/2005 06:32:51") returns 1.
Notes:	Compares two dates. Returns -1 if <Datetime1> is earlier than <Datetime2>. Returns 1 if <Datetime1> is later than <Datetime2>. Returns 0 if both dates are equal. Note that <Datetime> must be in the format MM/DD/YYYY HH:NN:SS.

DateFormat

Syntax:	DateFormat(<Datetime>, <Format>)
Parameters:	
<Datetime>	The date / time to format.
<Format>	Specifies the format to return.
Returns:	A date / time formatted according to the specification.
Examples:	DateFormat("01/10/2005 11:13:52", "M") returns “01/10/2005”.
Notes:	Reformats a Date/Time for display purposes. Valid format specifiers are: M: MM/DD/YYYY S: MM/DD/YY D: DD/MM/YYYY E: DD/MM/YY T: HH:NN:SS (24-hour) A: HH:NN:SS am/pm H: HH:NN (24-hour) P: HH:NN am/pm This function is for display only. Note that <Datetime> must be in the format MM/DD/YYYY HH:NN:SS.

Lookup Functions

CountCollect

Syntax:	CountCollect(<Field1>, <Value1>, ...)
Parameters:	
<Field1>	Prompt name of the field in the collected data file
<Value1>	Value to match
…	You may specify up to 5 Field/Value pairs
Returns:	Returns the number of matching records
Examples:	CountCollect(“sku”, “12345”) returns the number of records in the collected data file where the data collected for the prompt named “sku” is “12345”.
Notes:	Counts the number of records in the collected data file matching the specified criteria. If no match fields are specified, the total count of all collected records is returned.

DeleteCollect

Syntax:	DeleteCollect (<all> , <Field1>, <Value1>, ...)
Parameters:	
<all>	Pass “1” to delete all matching records, or “0” to delete just the first match.
<Field1>	The prompt name of the field in the collected data file to match
<Value1>	The value to match
…	Up to 5 Field/Value pairs may be specified
Returns:	Nothing
Examples:	DeleteCollect(“1”, “sku”, “12345”) deletes all collected data records where the value of the “sku” prompt is “12345”.
Notes:	At least one match field and value must be specified.

LastCollect

Syntax:	LastCollect(<Lookup>)
Parameters:	
<Lookup>	The prompt name of the field to retrieve
Returns:	Returns the last value collected for the prompt named <Lookup>
Examples:	LastCollect(“sku”) returns the last value collected for the prompt named “sku”.
Notes:	This function returns the last data collected.

LastCollectRecord

Syntax:	LastCollectRecord( )
Parameters:	None
Returns:	Returns the last collected data record.
Examples:	@record@ = LastCollectRecord() returns the last record collected.
Notes:	This function returns the last data collected.

LookupCollect

Syntax:	LookupCollect(<Lookup>, <Field1>, <Value1>, ...)
Parameters:	
<Lookup>	The Prompt Name of the field in the collected data file to lookup
<Field1>	The Prompt Name of a field in the collected data file to match
<Value1>	The data to match
…	Up to 5 Field/Value pairs may be specified
Returns:	Returns the value of a field in the collected data file
Examples:	LookupCollect(“qty”, “sku”, “12345”) returns the data collected for the prompt named “qty” where the value collected for the field named “sku” was “12345”.
Notes:	Performs a lookup from the collected data file. <Lookup> is the prompt name of the field to lookup. <Field1>, <Field2>, etc specify the prompt name of a field to match. <Value1>, <Value2>, etc specify the values of those fields in the collected data. Up to 5 match fields can be specified in this way. If more than one record matches the criteria specified, only the oldest first collected (oldest) record will be returned.

LookupCollectRecord

Syntax:	LookupCollectRecord(<Field1>, <Value1>, ...)
Parameters:	
<Field1>	The Prompt Name of a field in the collected data file to match
<Value1>	The data to match
…	Up to 5 Field/Value pairs may be specified
Returns:	Returns the entire collected data record matching the field / value pairs.
Examples:	@record@ = LookupCollectRecord(“sku”, “12345”) returns the collected data record where the value collected for the field named “sku” was “12345”.
Notes:	Performs a lookup from the collected data file. <Field1>, <Field2>, etc specify the prompt name of a field to match. <Value1>, <Value2>, etc specify the values of those fields in the collected data. Up to 5 match fields can be specified in this way. If more than one record matches the criteria specified, only the oldest first collected (oldest) record will be returned.

LookupCollectReverse

Syntax:	LookupCollectReverse(<Lookup>, <Field1>, <Value1>, ...)
Parameters:	
<Lookup>	The Prompt Name of the field in the collected data file to lookup
<Field1>	The Prompt Name of a field in the collected data file to match
<Value1>	The data to match
…	Up to 5 Field/Value pairs may be specified
Returns:	Returns the value of a field in the collected data file
Examples:	LookupCollectReverse(“qty”, “sku”, “12345”) returns the data collected for the prompt named “qty” where the value collected for the field named “sku” was “12345”.
Notes:	Performs a lookup from the collected data file. <Lookup> is the prompt name of the field to lookup. <Field1>, <Field2>, etc. specify the prompt name of a field to match. <Value1>, <Value2>, etc. specify the values of those fields in the collected data. Up to 5 match fields can be specified in this way. If more than one record matches the criteria specified, only the last collected (most recent) record will be returned.

LookupCollectReverseRecord

Syntax:	LookupCollectReverseRecord(<Field1>, <Value1>, ...)
Parameters:	
<Field1>	The Prompt Name of a field in the collected data file to match
<Value1>	The data to match
…	Up to 5 Field/Value pairs may be specified
Returns:	Returns the last collected data record matching the field/value pairs.
Examples:	@record@ = LookupCollectReverseRecord(“sku”, “12345”) returns the data collected where the value collected for the field named “sku” was “12345”.
Notes:	Performs a lookup from the collected data file. <Field1>, <Field2>, etc. specify the prompt name of a field to match. <Value1>, <Value2>, etc. specify the values of those fields in the collected data. Up to 5 match fields can be specified in this way. If more than one record matches the criteria specified, only the last collected (most recent) record will be returned.

LookupParseCollectField

Syntax:	LookupParseCollectField(<Field>, <Record>)
Parameters:	
<Field>	The field in the collected data record to return
<Record>	A stored record from the collected data file previously retrieved with the *LookupCollectRecord* function
Returns:	The data from the field <field> in the stored record
Examples:	LookupParseCollectField(“Prompt1.Textbox”, @Record@) returns the value in the “Prompt1.Textbox” field from the record stored in the user variable @Record@
Notes:	Parses a field from a collected data record. The field to parse should be given in <Field>. The data record should be in <Record> and be the result of the *LookupCollectRecord* or *LookupCollectRecordReverse* function.

LookupParseValidationField

Syntax:	LookupParseValidationField(<File>, <Field>, <Record>)
Parameters:	
<File>	The validation file to use for the lookup
<Field>	The field in the validation file to return
<Record>	A stored record from the validation file previously retrieved with the *LookupValidationRecord* function
Returns:	The data from the field <Field> in the stored record
Examples:	LookupParseValidationField(“val.txt”, “description”, @Record@) returns the value in the “description” field of the validation file “val.txt”, from the record stored in the user variable @Record@
Notes:	Parses a field from a validation file record. The filename without path should be in <File>. The validation file must have been defined in the *Validation Files* screen, even if it is not being used on the *Advanced Prompt Settings* screen. The field to parse should be given in <Field>. The data record should be in <Record> and be the result of the *LookupValidationRecord* function.

LookupValidation

Syntax:	LookupValidation(<File>, <Lookup>, <Field1>, <Value1>, ...)
Parameters:	
<File>	The validation file to use for the lookup
<Lookup>	The field in the validation file to return
<Field1>	The field to match
<Value1>	The value to match
…	Up to 5 Field/Value pairs may be specified
Returns:	The data from the field <Lookup> in the record matching the Field/Value criteria
Examples:	LookupValidation(“val.txt”, “description”, “sku”, “12345”) returns the value in the “description” field of the validation file “val.txt”, from the record where the “sku” field is equal to “12345”.
Notes:	Performs a lookup from a validation file. The filename without path should be in <File>. The field to lookup should be given in <Lookup>. The fields to match are given in <Field1>, <Field2>, etc. The value of each field is given in <Value1>, <Value2>, etc. Up to 5 match fields can be specified in this way. The validation file must have been defined in the *Validation Files* screen, even if it is not being used on the *Advanced Prompt Settings* screen.

LookupValidationRecord

Syntax:	LookupValidationRecord(<File>, <Field1>, <Value1>, ...)
Parameters:	
<File>	The validation file to use for the lookup
<Field1>	The field to match
<Value1>	The value to match
…	Up to 5 Field/Value pairs may be specified
Returns:	The record matching the Field/Value criteria
Examples:	@Record@ = LookupValidationRecord("val.txt", "sku", "12345") returns the record (all the fields) from the validation file “val.txt” where the “sku” field is equal to “12345”.
Notes:	Performs a lookup from a validation file and returns the entire record. The filename without path should be in <file>. The fields to match are given in <Field1>, <Field2>, etc. The value of each field is given in <Value1>, <Value2>, etc. Up to 5 match fields can be specified in this way. The validation file must have been defined in the *Validation Files* screen, even if it is not being used on the *Advanced Prompt Settings* screen. Use the *LookupParseValidationField* function to parse individual fields from the returned record.

PickListField

Syntax:	PickListField(<Field>)
Parameters:	
<Field>	The name of the field to return
Returns:	The data from a field in a validation file
Examples:	PickListField(“descr”) returns the value of the “descr” field in the validation file for this record.
Notes:	This function is used only in the PickList In-Prompt Script. As each record is added to the picklist, the PickList In-Prompt Script is called. You can format the text to be displayed in the picklist. This function can be used to retrieve the value of a validation file field.

SaveCollectedData

Syntax:	SaveCollectedData( )
Parameters:	None
Returns:	Nothing
Examples:	SaveCollectedData ( ) saves the current values of all prompts to the collected data file.
Notes:	This function causes a collected data record to be written, using the current values of all prompts and elements. This is exactly like accepting the last prompt in the program. This function is only support on Windows CE and PocketPC clients, and the PC Client. It is not supported on DOS devices.

SumCollect

Syntax:	SumCollect (<Lookup>, <Field1>, <Value1>, …)
Parameters:	
<Lookup>	The prompt name of the field in the collected data file to sum
<Field1>	The prompt name of the field in the collected data file to match
<Value1>	The value to match
…	Up to 5 Field/Value pairs may be specified
Returns:	Returns the sum of the specified field matching the criteria
Examples:	SumCollect(“qty”, “sku”, “12345”) returns the sum of the data collected for the “qty” field in the collected data file where the data collected for the prompt named “sku” is “12345”.
Notes:	Sums the values in the collected data file matching the specified criteria. <Lookup> specifies the prompt name of the field in the collected data file to sum. <Field1>, <Field2> etc specify the prompt name of the field in the collected data file to match. <Value1>, <Value2> etc specify the values to match. If no match fields are specified, the sum for all collected records is returned.

UpdateCollect

Syntax:	UpdateCollect (<Field>, <Data>, <All>, <Field1>, <Value1>, ...)
Parameters:	
<Field>	The name of the prompt in the collected data to update
<Data>	The data to store as the new value.
<All>	Pass “1” to update all matching records, or “0” to update just the first match.
<Field1>	The prompt name of the field in the collected data file to match
<Value1>	The value to match
	Up to 5 Field/Value pairs may be specified
Returns:	Nothing
Examples:	UpdateCollect(“qty”, “10”, “1”, “sku”, “12345”) updates all collected data records where the value of the “sku” prompt is “12345”, and sets the “qty” field to “10”.
Notes:	At least one Match field and Value must be specified.

UpdateCollectField

Syntax:	UpdateCollectField (<record> , <field>, <data>)
Parameters:	
<record>	The collected data record.
<field>	The name of the field to update.
<data>	The data to place into the field.
Returns:	Nothing
Examples:	UpdateCollect(@record@, “qty”, “10”) updates the collected data record in @record@, setting the “qty” field to “10”.
Notes:	This function updates a collected data record in a string variable. This function is used in conjunction with *LookupCollectRecord* or *LookupCollectRecordReverse* to allow changing the values of any number of fields on a collected data record. You must call *UpdateCollectRecord* to write the record back to the collected data file.

UpdateCollectRecord

Syntax:	UpdateCollectRecord (<record>, <All> , <Field1>, <Value1>, …)
Parameters:	
<record>	The collected data in a string variable.
<All>	Pass “1” to update all matching records, or “0” to update just the first match.
<Field1>	The prompt name of the field in the collected data file to match
<Value1>	The value to match
…	Up to 5 Field/Value pairs may be specified
Returns:	Nothing
Examples:	UpdateCollectRecord(@record@, “1”, “sku”, “12345”) updates the collected data records where the value of the “sku” prompt is “12345”.
Notes:	At least one Match field and Value must be specified. After retrieving a collected data record into a string with *LookupCollectRecord* or *LookupCollectRecordReverse*, use *UpdateCollectField* to change the values. Finally, use *UpdateCollectRecord* to save those changes back to the collected data file.

Response Functions

ResponseSource

Syntax:	ResponseSource()
Parameters:	None
Returns:	Returns the Source of the last input. Valid sources are srcKeyboard, srcScanner, srcImage, or srcText.
Examples:	ResponseSource() returns srcScanner if the last input was scanned.
Notes:	Returns the Source of the last input.

ResponseSymbology

Syntax:	ResponseSymbology()
Parameters:	None
Returns:	Returns the Symbology of the barcode scanned for the last response.
Examples:	ResponseSymbology() returns 1 (bcCode39) if the last barcode scanned was Code 39.
Notes:	Returns the barcode symbology of the last scan. See the Constants list for the symbology IDs.

ValidationFail

Syntax:	ValidationFail(<String> , [element])
Parameters:	
<String>	The error message to be displayed on the terminal display
[element]	Optional. The name of the element to receive the keyboard focus.
Returns:	Nothing
Examples:	ValidationFail("Error", “prompt1.textbox1”)
Notes:	This function may be used in the After Prompting and After Validation scripts. Call to cause the validation of this response to fail. The terminal will display the message you place in <String>. If this function is called multiple times in a script, the message from the first occurrence will be displayed. If a multi-prompt element name is specified in [element], the keyboard focus will be set to this element when the notification message is closed. If the <String> parameter is an empty string, no message will be displayed but the validation will still fail, and the focus will be set to the element specified in [element].

Notification Functions

Beep

Syntax:	Beep()
Parameters:	None
Returns:	Nothing
Examples:	Beep()
Notes:	Causes the terminal to beep a high-pitched beep.

Buzz

Syntax:	Buzz()
Parameters:	None
Returns:	Nothing
Examples:	Buzz()
Notes:	Causes the terminal to buzz a low-pitched buzz.

EnableCentering

Syntax:	EnableCentering(<Size>)
Parameters:	
<Size>	Size of the region around the aimer to locate symbols
Returns:	Nothing
Examples:	EnableCentering(20)
Notes:	Enables Centering for a HandHeld Products Imager. This restricts the area in which the imager will locate barcodes. This function is only supported on imager-equipped terminals. Setting <Size> to zero disables centering.

EnableAimer

Syntax:	EnableAimer(<Time>)
Parameters:	
<Time>	Specifies the length of time to show the aiming dot, in tenths of a second.
Returns:	Nothing
Examples:	EnableAimer(50)
Notes:	Enables the Laser Aimer on laser-equipped scanners that support an aiming dot. This turns on the laser for a short time before beginning to sweep, to help locate the laser on the barcode.

EnableALD

Syntax:	EnableALD(<Size>)
Parameters:	
<Size>	Size of the region around the aimer to locate symbols
Returns:	Nothing
Examples:	EnableALD(1)
Notes:	Enables Advanced Linear Decoding for a HandHeld Products Imager. This restricts the area in which the imager will locate barcodes. This function is only supported on imager-equipped terminals. Setting <Size> to zero disables ALD.

ExitProgram

Syntax:	ExitProgram( )
Parameters:	None
Returns:	Nothing
Examples:	ExitProgram( )
Notes:	Causes the data collection program to exit and return to the Client's main menu. Supported for PocketPC Devices, CE Devices, and PC Client. Not supported on DOS Devices.

FlashLEDs

Syntax:	FlashLEDs()
Parameters:	None
Returns:	Nothing
Examples:	FlashLEDs()
Notes:	Causes the terminal to flash the SCAN and DECODE LEDs for ½ second.

GetBatteryLife

Syntax:	GetBatteryLife( )
Parameters:	None
Returns:	Returns the remaining battery life percentage from 0 to 100.
Examples:	@ret@ = GetBatteryLifeStatus( )
Notes:	If the battery change status cannot be determined, this function returns -1, or on some devices, 255. The Simulator and PC Client always return -1.

GetPowerStatus

Syntax:	GetPowerStatus( )
Parameters:	none
Returns:	Returns 1 if external power is applied, or 0 if not.
Examples:	@ret@ = GetPowerStatus( )
Notes:	Determine whether the portable device is being powered by a cradle or charge cable.

GoToPrompt

Syntax:	GoToPrompt(<Prompt> , <Validate>)
Parameters:	
<Prompt>	The name of the prompt to move to next.
<Validate>	Whether to validate the prompt before advancing to the prompt specified.
Returns:	Nothing
Examples:	GoToPrompt (“Prompt4” , 1)
Notes:	Causes the data collection program to go to the specified prompt. If <Validate> is not zero, the data for the prompt will be saved into the elements, and they will be validated. If the validation fails, the program will not move to the new prompt. Supported for PocketPC Devices, CE Devices, and PC Client. Not supported on DOS Devices.

Message

Syntax:	Message(<Message> , <Caption> , <Button1> , <Button2>)
Parameters:	
<Message>	The message text to display
<Caption>	The window caption to display
<Button1>	Text for the first button
<Button2>	Text for the second button. The 2nd button may be omitted by using the empty string (“”) for the <Button2> text.
Returns:	Either a 1 or a 2 depending on which button is pressed (or clicked)
Examples:	@Data@ = Message("Are you sure?", "Confirm", "<Yes>", "<No>")
Notes:	Displays a message to the user. The message text is in <Message>, and the window caption is in <Caption>. You may specify text for one or two buttons using <Button1> and <Button2>. If <Button2> is an empty string, only <Button1> will be displayed, and it will be centered. This function returns 1 or 2 corresponding to whether button1 or button2 is pressed.

PlaySound

Syntax:	PlaySound(<Soundfile>)
Parameters:	
<Soundfile>	The name of a WAV file to play
Returns:	Nothing
Examples:	PlaySound(“sound.wav”)
Notes:	Causes the terminal to play a WAV file. Supported on the Windows CE and Pocket PC devices. The WAV file must be in the program directory, or the terminal will simply beep.

SetPowerDownMode

Syntax:	SetPowerDownMode(<mode>)
Parameters:	
<mode>	The powerdown mode to set
Returns:	Nothing
Examples:	SetPowerDownMode(1)
Notes:	Sets the device power down mode for Windows CE or PocketPC/Windows Mobile devices. The possible modes are: 0=Standard. The device suspends after a period of inactivity. 1=Unattended. The device will turn off the screen and buttons, but programs will continue to run. On most devices, the GPRS and Bluetooth radios will continue to run, but WiFi is usually powered down. 2=Always On. The device will stay on and running, and will not suspend. In all cases, this mode stays in effect until the Data Collection program is exited back to the Main Menu, where the Standard mode is set.

Print/Other Functions

DownloadData

Syntax:	DownloadData(<program>)
Parameters:	
<Filename>	The ITB file whose data should be downloaded.
Returns:	Nothing
Examples:	DownloadData ("program.itb")
Notes:	Downloads the collected data through a serial, USB or Activesync connection. This function reproduces what the Send Data to PC button on the Main Menu does. This function connects to the Autodownload Server only. It does not connect to the OMNI Server.

FileAppend

Syntax:	FileAppend(<Filename> , <String>)
Parameters:	
<Filename>	File to append to
<String>	Data to be appended to file
Returns:	Nothing
Examples:	FileAppend("File.txt" , "This is a test" & ascCR & ascLF)
Notes:	Appends the data in <String> to the file named <Filename>. CR/LF are not appended automatically. You can append them using the constants ascCR and ascLF.

FileCopy

Syntax:	FileCopy(<sourcefile>, <destfile>, <overwrite>)
Parameters:	
<sourcefile>	Original file to copy.
<destfile>	Name of the destination file to create.
<overwrite>	1 to overwrite an existing destination file, 0 to fail if the destination file exists.
Returns:	This function returns 1 if the file copy succeeds, or 0 if it fails.
Examples:	FileCreate("File.txt")
Notes:	Copies the file named in <sourcefile> to <destfile>. Overwrites an existing <destfile> if the <overwrite> parameter is not zero, otherwise this function fails if the destination file already exists.

FileCreate

Syntax:	FileCreate(<Filename>)
Parameters:	
<Filename>	File to create
Returns:	Nothing
Examples:	FileCreate("File.txt")
Notes:	Creates an empty file named <Filename>.

FileDelete

Syntax:	FileDelete(<Filename>)
Parameters:	
<Filename>	File to delete
Returns:	Nothing
Examples:	FileDelete("Text.txt")
Notes:	Deletes the file named by <Filename>.

FileExists

Syntax:	FileExists(<Filename>)
Parameters:	
<Filename>	File to check
Returns:	1 if the file exists, 0 if not.
Examples:	@ret@ = FileExists("Text.txt")
Notes:	Checks for the file named by <Filename>.

FileRename

Syntax:	FileRename(<Oldname>, <Newname>)
Parameters:	
<Oldname>	File to rename
<Newname>	New name for file
Returns:	Nothing
Examples:	FileRename("OldFile.txt","NewFile.txt")
Notes:	Renames the file named by <Filename> to <Newname>.

IrDAFile

Syntax:	IrDAFile(<Filename>)
Parameters:	
<Filename>	File on terminal to send to IrDA port, typically to a printer
Returns:	Returns 1 if the function was successful, returns 0 if function failed
Examples:	IrDAFile("File.prn")
Notes:	Sends the contents of the disk file named <Filename> to the IrDA printer. No translations or substitutions are performed. If the connection is not able to be made, the Retry/Cancel message will be displayed to the user. The printing functions do not do anything in the simulator.

IrDAPrint

Syntax:	IrDAPrint(<Printfile>)
Parameters:	
<Printfile>	File containing printer-specific commands, typically generated in a Label Design package. The PrintFile is defined in the *Print Files* screen.
Returns:	Returns 1 if the function was successful, returns 0 if function failed
Examples:	IrDAPrint("PrintFile")
Notes:	Sends <Printfile>, defined in the *Print Files* screen, to the IrDA printer. Performs variable substitution before sending. If the connection is not able to be made, the Retry/Cancel message will be displayed to the user. The printing functions do not do anything in the simulator.

IrDAString

Syntax:	IrDAString(<String>)
Parameters:	
<String>	String to send to the IrDA port (typically to a printer)
Returns:	Returns 1 if the function was successful, returns 0 if function failed
Examples:	IrDAString("ABCDEFG")
Notes:	Sends the string to the IrDA port which would typically be sent to a IrDA printer. If the connection is not able to be made, the Retry/Cancel message will be displayed to the user. The printing functions do not do anything in the simulator.

LoadProgram

Syntax:	LoadProgram(<program>)
Parameters:	
<Filename>	The ITB file to be uploaded.
Returns:	Nothing
Examples:	LoadProgram ("program.itb")
Notes:	Uploads the ITB file and all associated validation and support files through a serial, USB or Activesync connection. This function reproduces what the Load Program button on the Main Menu does. This function connects to the Autodownload Server only. It does not connect to the OMNI Server.

RFPrtFile

Syntax:	RFPrtFile(<Address>, <Port>, <Filename>)
Parameters:	
<Address>	Printer IP Address
<Port>	Port to use
<Filename>	File on terminal to send
Returns:	Returns 1 if the function was successful, returns 0 if function failed
Examples:	RFPrtFile("192.168.1.1", "6101", "File.prn")
Notes:	Sends the contents of the disk file named <Filename> to an RF printer. No translations or substitutions are performed. The printer IP Address should be in <Address> and the port in <Port>. If the RF connection is not able to be made, the Retry/Cancel message will be displayed to the user. The printing functions do not do anything in the simulator.

RFPrtPrint

Syntax:	RFPrtPrint(<Address>, <Port>, <PrintFile>)
Parameters:	
<Address>	Printer IP Address
<Port>	Port to use
<PrintFile>	File containing printer-specific commands, typically generated in a Label Design package. The printfile is defined in the *Print Files* screen.
Returns:	Returns 1 if the function was successful, returns 0 if function failed
Examples:	RFPrtPrint("192.168.1.1", "6101", "PrintFile")
Notes:	Sends <PrintFile>, defined in the *Print Files* screen, to an RF printer. Performs variable substitution before sending. The printer IP Address should be in <Address> and the port in <Port>. If the RF connection is not able to be made, the Retry/Cancel message will be displayed to the user. The printing functions do not do anything in the simulator.

RFPrtString

Syntax:	RFPrtString(<Address>, <Port>, <String>)
Parameters:	
<Address>	Printer IP Address
<Port>	Port to use
<String>	String to send
Returns:	Returns 1 if the function was successful, returns 0 if function failed
Examples:	RFPrtString("192.168.1.1", "6101", "ABCDEFG")
Notes:	Sends <String> to an RF printer. The printer IP Address should be in <Address> and the port in <Port>. If the RF connection is not able to be made, the Retry/Cancel message will be displayed to the user. The printing functions do not do anything in the simulator.

SerialClose

Syntax:	SerialClose(<Port>)
Parameters:	
<Port>	Number indicating the serial port to close. This can be in the range serCOM1 to serCOM9.
Returns:	Nothing
Examples:	SerialClose(serCOM1)
Notes:	Closes the serial port previously opened with SerialOpen.

SerialFlush

Syntax:	SerialFlush(<Port>)
Parameters:	
<Port>	Number indicating the serial port to flush. This can be in the range serCOM1 to serCOM9.
Returns:	Nothing
Examples:	SerialFlush (serCOM1)
Notes:	Flushes any received data that has not been read from the port.

SerialOpen

Syntax:	SerialOpen(<Port>, <baud>, <parity>, <bits>)
Parameters:	
<Port>	Number indicating the serial port to open. This can be in the range serCOM1 to serCOM9.
<baud>	The baud rate to use.
<parity>	Indicate Odd, Even, or No parity.
<bits>	The number of data bits to use for communications.
Returns:	Returns 1 if the port was opened, or 0 if not.
Examples:	@ret@ = SerialOpen(serCOM1, serBaud9600, serParityNone, serData8)
Notes:	Once the port has been opened, it will remain open until closed by SerialClose or until the data collection program is exited.

SerialPrtFile

Syntax:	SerialPrtFile(<Port>, <Baud>, <Parity>, <Bits>, <Filename>)
Parameters:	
<Port>	Number indicating the serial port to print to
<Baud>	Baud rate
<Parity>	Parity setting
<Bits>	Number of data bits
<Filename>	File on terminal to send to serial port, typically to a printer
Returns:	Returns 1 if the function was successful, returns 0 if function failed
Examples:	SerialPrtFile(serCOM1, SerBaud9600, serParityNone, serData8, "File.prn")
Notes:	Sends the contents of the disk file named <Filename> to a Serial printer. No translations or substitutions are performed. The serial port is in <Port>, the baud rate in <Baud>, the parity in <Parity> and data bits in <Bits>. If the serial connection is not able to be made, the Retry/Cancel message will be displayed to the user.

SerialPrtPrint

Syntax:	SerialPrtPrint(<Port>, <Baud>, <parity>, <Bits>, <PrintFile>)
Parameters:	
<Port>	Number indicating the serial port to print to
<Baud>	Baud rate
<Parity>	Parity setting
<Bits>	Number of data bits
<PrintFile>	File containing printer-specific commands, typically generated in a Label Design package. The PrintFile is defined in the *Print Files* screen.
Returns:	Returns 1 if the function was successful, returns 0 if function failed
Examples:	SerialPrtPrint(serCOM1, SerBaud9600, serParityNone, serData8, "PrintFile")
Notes:	Sends <Printfile>, defined in the *Print Files* screen, to a Serial printer. Performs variable substitution before sending. The serial port is in <Port>, the baud rate in <Baud>, the parity in <Parity> and data bits in <Bits>. If the serial connection is not able to be made, then a Retry/Cancel message will be displayed to the user.

SerialPrtString

Syntax:	SerialPrtString(<Port>, <Baud>, <Parity>, <Bits>, <String>)
Parameters:	
<Port>	Number indicating the serial port to print to
<Baud>	Baud rate
<Parity>	Parity setting
<Bits>	Number of data bits
<String>	String to send to the serial port (typically to a printer)
Returns:	Returns 1 if the function was successful, returns 0 if function failed
Examples:	SerialPrtString(serCOM1, serBaud9600,serParityNone, serData8,"ABCDEFG")
Notes:	Sends <String> to a Serial printer. The serial port is in <Port>, the baud rate in <Baud>, the parity in <Parity> and data bits in <Bits>. If the serial connection is not able to be made, then a Retry/Cancel message will be displayed to the user.

SerialRead

Syntax:	SerialRead(<Port> , <Baud> , <Parity> , <Bits> , <Timeout> , <Endofline>)
Parameters:	
<Port>	Number indicating the COM port to read
<Baud>	Baud rate
<Parity>	Parity setting
<Bits>	Number of data bits
<Timeout>	The terminal will wait <Timeout> milliseconds before timing out.
<Endofline>	Data from the port will be read until the <Endofline> is read
Returns:	The data that is read from the port
Examples:	@Data@ = SerialRead(serCOM1, serBaud9600, serParityNone, serData8, 5000, ascCR)
Notes:	Writes the data in <String> to <Port> using the <Baud>, <Parity> and <Bits> specified. Non-printable ASCII characters will be encoded as [XXX] where XXX is the 3-digit decimal code for the character. The terminal will wait <Timeout> milliseconds before timing out. A timeout of 0 will cause the timeout to be infinite. A Timeout between 1 and 100ms will not display the status message while waiting. This allows silent polling of the serial port by a timer function. The Escape key will terminate the function.

SerialWrite

Syntax:	SerialWrite(<Port> , <Baud> , <Parity> , <Bits> , <Timeout> , <String>)
Parameters:	
<Port>	Number indicating the COM port
<Baud>	Baud rate
<Parity>	Parity setting
<Bits>	Number of data bits
<Timeout>	The terminal will wait <Timeout> milliseconds before timing out.
<String>	Data to be written to the serial port
Returns:	The number of characters written to the port
Examples:	@Written@ = SerialWrite(serCOM1, SerBaud9600, serParityNone, serData8, 1000, "12345" & ascCR & ascLF)
Notes:	Writes the data in <String> to <Port> using the <Baud>, <Parity> and <Bits> specified. Non-printable ASCII characters can be encoded as [XXX] where XXX is the 3-digit decimal code for the character.

SerialWriteRead

Syntax:	SerialWriteRead(<Port> , <Baud> , <Parity> , <Bits> , <Timeout> , <String>, <EndOfLine>)
Parameters:	
<Port>	Number indicating the COM port
<Baud>	Baud rate
<Parity>	Parity setting
<Bits>	Number of data bits
<Timeout>	The terminal will wait <Timeout> milliseconds before timing out.
<String>	Data to be written to the serial port
<EndOfLine>	Data from the port will be read until the <Endofline> is read
Returns:	The data read from the port
Examples:	@Data@ = SerialWriteRead(serCOM1, serBaud9600, serParityNone, serData8, 1000, "12345" & ascCR & ascLF & ascCR)
Notes:	Writes the data in <String> to <Port> using the <Baud>, <Parity> and <Bits> specified. Then, reads the port until the character specified in <Endofline> is read. The terminal will wait <Timeout> milliseconds before timing out. A timeout of 0 will cause the timeout to be infinite. A Timeout between 1 and 100ms will not display the status message while waiting. This allows silent polling of the serial port by a timer function. The Escape key will terminate the function. This function returns the data read from the port. Non-printable ASCII characters can be encoded as [XXX] where XXX is the 3-digit decimal code for the character. Returned non-printable ASCII data will also be encoded.

Shell

Syntax:	Shell(<CommandLine>, [NoWait])
Parameters:	
<CommandLine>	Command line for external program to run
[NoWait]	Optional parameter. Pass 1 if you do not want to wait for a result. 0 or no parameter will wait for the result.
Returns:	Data in the itscript.ret file, if waiting for a response.
Examples:	Shell("MyProgram.exe")
Notes:	Shells to an external program. You may specify command line and parameters. The external program should write the data to be returned in a file named "itscript.ret". The Shell function will read that data and return it.

ShowSIP

Syntax:	ShowSIP(<show>)
Parameters:	
<show>	Specifies whether to show or hide the Soft Input Panel.
Returns:	Nothing
Examples:	ShowSIP(1)
Notes:	Show or hide the Pocket PC Soft Input Panel. The <show> parameter specifies whether to show or hide the panel. Specify 1 or TRUE to show the panel, and 0 or FALSE to hide it. This function only works for devices that support the Soft Input Panel.

Exec

Syntax:	Exec(<text>)
Parameters:	
<text>	Script code to execute
Returns:	Nothing
Examples:	Exec("@Test@ = Left("ABCDE", 2)")
Notes:	Executes a single line of script code.

GlobalScript

Syntax:	GlobalScript(<scriptname>)
Parameters:	
<scriptname>	The name of the global script to execute.
Returns:	Returns 1 if the global script was executed, or 0 if it could not be found
Examples:	GlobalScript("TestScript")
Notes:	Executes a Global Script. The <scriptname> parameter specifies a script defined on the Global Scripts screen.

GlobalScriptFile

Syntax:	GlobalScriptFile(<scriptname>, <filename>)
Parameters:	
<scriptname>	The name of the global script to execute.
<filename>	The name of the external file containing the global scripts.
Returns:	Returns 1 if the global script was executed, or 0 if it could not be found
Examples:	GlobalScriptFile("TestScript", “Script.txt”)
Notes:	Executes a Global Script from an external file. The <scriptname> parameter specifies the name of the script, and the <filename> is the name of the external file. The file must be located in the same directory as the ITB file.

GPSOpen

Syntax:	GPSOpen( )
Parameters:	None
Returns:	Nothing. The GPS driver returns immediately while the hardware is initialized.
Examples:	GPSOpen( )
Notes:	Opens a connection to the GPS device. This connection can take a long time to begin functioning, so use the GPSIsOpen function to poll the driver to determine when the hardware is running. Once the GPS device is opened, it stays open until GPSClose is called, or until the current data collection program is exited. This function is supported only on Windows Mobile 5 (or higher) devices with the Microsoft GPS Intermediate Driver.

GPSIsOpen

Syntax:	GPSIsOpen( )
Parameters:	None
Returns:	0 if the device is not ready, 1 if it is.
Examples:	@ret@ = GPSIsOpen( )
Notes:	This function queries the status of the GPS driver to determine when the hardware is running. It can take several seconds or longer after calling GPSOpen for the hardware to report that it is ready. This function is supported only on Windows Mobile 5 (or higher) devices with the Microsoft GPS Intermediate Driver.

GPSClose

Syntax:	GPSClose( )
Parameters:	None
Returns:	Nothing.
Examples:	GPSClose( )
Notes:	Closes an open connection to the GPS device. This function is supported only on Windows Mobile 5 (or higher) devices with the Microsoft GPS Intermediate Driver.

GPSGetPosition

Syntax:	GPSGetPosition(<maxage>, <latitude>, <longitude>, <altitude>, <speed>, <heading>, <numsatellites>)
Parameters:	
<maxage>	The maxmimum age of valid data, in seconds.
<latitude>	A user variable that the Latitude value will be returned in.
<longitude>	A user variable that the Longitude value will be returned in.
<altitude>	A user variable that the Altitude value will be returned in.
<speed>	A user variable that the Speed value will be returned in.
<heading>	A user variable that the Heading value will be returned in.
<numsatellites>	A user variable that the Number of Satellites value will be returned in.
Returns:	0 on error or if no fix is obtained, 2 for a 2-d fix, 3 for a 3-d fix.
Examples:	@fix@ = GPSGetPosition(30, @Lat@, @Long@, @Alt@, @Speed@, @Heading@, @NumSat@)
Notes:	Queries the GPS device for the current position. If the GPS device has calculated valid data within the time specified in <maxage>, it will be returned and placed in the user variables specified as parameters. If the device does not have valid data for one or more values, the most recent valid data will be returned with an '*' as the first character. For example, *49.12345. Use this '*' character to know that the data may not be accurate. Note that some fields may be current while others are not. Use the <numsatellites> and the return value to gauge the reliability of the position. At least 3 satellites are required to get an accurate latitude/longitude (2-D fix), while at least 4 are required to get an accurate altitude (3-D fix). This function is supported only on Windows Mobile 5 (or higher) devices with the Microsoft GPS Intermediate Driver.

Multiprompt Functions

AddItem

Syntax:	AddItem (<element>, <text>, <data>)
Parameters:	
<element>	The name of the element to check.
<text>	The text to insert into the list.
<data>	The Item Data to use for the new list item.
Returns:	Nothing
Examples:	AddItem ("Prompt1.ComboBox", "New Item", "123")
Notes:	For Combo Boxes and List Boxes, adds the item with <text> and <data> to the end of the list.

Clear

Syntax:	Clear(<Elementname>)
Parameters:	
<Elementname>	The name of the element to clear.
Returns:	Nothing
Examples:	Clear("prompt.element")
Notes:	Clears the multiprompt element. Elements are referenced using the syntax “prompt.element”.

DeleteItem

Syntax:	DeleteItem (<element>, <index>)
Parameters:	
<element>	The name of the element to check.
<index>	The index of the item to delete.
Returns:	Nothing
Examples:	DeleteItem ("Prompt1.ComboBox", 2)
Notes:	For Combo Boxes, List Boxes and Grids, deletes the item at position <index> from the list.

Disable

Syntax:	Disable(<Elementname>)
Parameters:	
<Elementname>	The name of the element to disable.
Returns:	Nothing
Examples:	Disable("prompt.element")
Notes:	Disables a multiprompt element that had been enabled. Elements are referenced using the syntax “prompt.element”.

Enable

Syntax:	Enable(<Elementname>)
Parameters:	
<Elementname>	The name of the element to enable.
Returns:	Nothing
Examples:	Enable("prompt.element")
Notes:	Enables a multiprompt element that had been disabled. Elements are referenced using the syntax “prompt.element”.

FindIndexByData

Syntax:	FindIndexByData(<Elementname>, <data>)
Parameters:	
<Elementname>	The name of the element to enable.
<data>	The data to find in the list.
Returns:	The index of the first matching item.
Examples:	FindIndexByData("prompt.element", “1234”)
Notes:	For Combo Boxes and List Boxes, returns the index of the item which has the item data matching <data>. The search is case-insensitive. The first row is number 1, not 0.

FindIndexByText

Syntax:	FindIndexByText(<Elementname>, <text>)
Parameters:	
<Elementname>	The name of the element to enable.
<text>	The text to find in the list.
Returns:	The index of the first matching item.
Examples:	FindIndexByText("prompt.element", “ABCD”)
Notes:	For Combo Boxes and List Boxes, returns the index of the item which has the text matching <text>. The search is case-insensitive. The first row is number 1, not 0.

GetCount

Syntax:	GetCount (<element>)
Parameters:	
<element>	The name of the element to check.
Returns:	The number of items in the list.
Examples:	GetCount ("prompt.element")
Notes:	For Combo Boxes, List Boxes and Grids, returns the number of items in the list.

GetIndex

Syntax:	GetIndex(<element>)
Parameters:	
<element>	The name of the element whose index should be retrieved.
Returns:	The index of the currently selected item
Examples:	GetIndex ("prompt.element")
Notes:	For Combo Boxes, List Boxes, Grids, and Multi Lists, returns the index of the currently selected row. The first row is number 1, not 0. For Text Boxes, returns the position of the cursor in the Text Box. The left of the first character position is 0.

GetItemData

Syntax:	GetItemData (<Elementname>, <index>)
Parameters:	
<Elementname>	The name of the element to search.
<index>	The index of the item to retrieve.
Returns:	The item data of the matching item.
Examples:	@text@ = GetITemData ("prompt.element", 2)
Notes:	For Combo Boxes and List Boxes, retrieves the data of the item specified by <index>. The first row is number 1, not 0.

GetItemText

Syntax:	GetItemText(<Elementname>, <text>)
Parameters:	
<Elementname>	The name of the element to search.
<text>	The text to find in the list.
Returns:	The text of the first matching item.
Examples:	@text@ = GetItemText ("prompt.element", “ABCD”)
Notes:	For Combo Boxes and List Boxes, retrieves the text of the item specified by <index>. The first row is number 1, not 0.

Hide

Syntax:	Hide(<Elementname>)
Parameters:	
<Elementname>	The name of the element to hide.
Returns:	Nothing
Examples:	Hide("prompt.element")
Notes:	Hides a multiprompt element that had been shown. Elements are referenced using the syntax “prompt.element”.

InsertItem

Syntax:	InsertItem (<element>, <index>, <text>, <data>)
Parameters:	
<element>	The name of the element to check.
<index>	The index of the item to insert.
<text>	The text to insert into the list.
<data>	The Item Data to use for the new list item.
Returns:	Nothing
Examples:	InsertItem ("Prompt1.ComboBox", "2", "New Item", "123")
Notes:	For Combo Boxes and List Boxes, inserts the item with <text> and <data> at the position <index>. Position 0 is before the first item, position 1 is after the first item.

IsEnabled

Syntax:	IsEnabled(<Elementname>)
Parameters:	
<Elementname>	The name of the element to check.
Returns:	1 if the element is enabled, or 0 if not enabled.
Examples:	@ret@ = IsEnabled("prompt.element")
Notes:	Checks to see if the multiprompt element is enabled or disabled. Elements are referenced using the syntax “prompt.element”.

IsVisible

Syntax:	IsVisible(<Elementname>)
Parameters:	
<Elementname>	The name of the element to check.
Returns:	1 if the element is visible, or 0 if not visible.
Examples:	@ret@ = IsVisible("prompt.element")
Notes:	Checks to see if the multiprompt element is visible or hidden. Elements are referenced using the syntax “prompt.element”.

Keypress

Syntax:	Keypress( )
Parameters:	None
Returns:	Returns the character code of the key pressed to trigger the event.
Examples:	@Key@ = Keypress( )
Notes:	Valid only during the OnKeyPress Event. Calling this function during any other script or event is undefined.

Refresh

Syntax:	Refresh(<Elementname>)
Parameters:	
<Elementname>	The name of the element to refresh.
Returns:	Nothing
Examples:	Refresh("prompt.element")
Notes:	Causes a multiprompt element to be refreshed. This causes initialization scripts to be re-run and redisplays the element. Elements are referenced using the syntax “prompt.element”.

RGB

Syntax:	RGB(<Red>, <Green>, <Blue>)
Parameters:	
<Red>	The decimal value of the Red component of the color.
<Green>	The decimal value of the Green component of the color.
<Blue>	The decimal value of the Blue component of the color.
Returns:	A numeric value indicating the complete RGB value.
Examples:	RGB(255, 0, 0)
Notes:	Creates an RGB value from the component colors. This can be used in any of the custom color scripts.

Select

Syntax:	Select(<Elementname>, <Index>)
Parameters:	
<Elementname>	The name of the element to select.
<Index>	Controls the selection for each element type. For Text Input boxes, an <Index> of 0 removes the selection, while an <Index> of 1 selects all text. For Comboboxes, Grids, and Listboxes, the <Index> specifies the number of the row to select. The first row number is 1, not 0.
Returns:	Nothing
Examples:	Select("prompt.element")
Notes:	Selects an item in a list, or selects the text in a Text Input box. Elements are referenced using the syntax “prompt.element”.

SetIndex

Syntax:	SetIndex(<element>, <index>)
Parameters:	
<Elementname>	The name of the element whose index should be set.
<index>	The index that should be selected.
Returns:	Nothing
Examples:	SetIndex ("prompt.element", 2)
Notes:	For Combo Boxes, List Boxes, Grids, and Multi Lists, sets the currently selected row to the index specified. The first row is number 1, not 0. For Text Boxes, sets the position of the cursor in the Text Box. The first character position is 0. Setting the cursor position removes any selection.

SetItemData

Syntax:	SetItemData(<element>, <index>, <data>)
Parameters:	
<Elementname>	The name of the element whose index should be set.
<index>	The index that should be selected.
<data>	The item data that should be placed into the list.
Returns:	Nothing
Examples:	SetItemData ("prompt.element", 2, “1234”)
Notes:	For Combo Boxes and List Boxes, sets the data of the item specified by <index>. The first row is number 1, not 0.

SetItemText

Syntax:	SetItemText(<element>, <index>, <text>)
Parameters:	
<Elementname>	The name of the element whose index should be set.
<index>	The index that should be selected.
<text>	The text that should be placed into the list.
Returns:	Nothing
Examples:	SetItemText ("prompt.element", 2, “ABCD”)
Notes:	For Combo Boxes and List Boxes, sets the text of the item specified by <index>. The first row is number 1, not 0.

SetFocus

Syntax:	SetFocus(<Elementname>)
Parameters:	
<Elementname>	The name of the element to receive the keyboard focus.
Returns:	Nothing
Examples:	SetFocus("prompt.element")
Notes:	Sets the keyboard focus to the multiprompt element. That element will now receive keyboard input. Elements are referenced using the syntax “prompt.element”.

Show

Syntax:	Show(<Elementname>)
Parameters:	
<Elementname>	The name of the element to show.
Returns:	Nothing
Examples:	Show("prompt.element")
Notes:	Shows a multiprompt element that had been hidden. Elements are referenced using the syntax “prompt.element”.

Keywords

IF

Syntax:	IF(<expression>)
Parameters:	
<expression>	Logical expression to be evaluated

Examples:	IF(@UserVar@ = 1)
	  @Count@ = @Count@ + 1
	ENDIF

Notes:	An IF Statement. This can take the form:
	IF(<expression>)
	  Statements
	ELSEIF(<expression>)
	  Statements
	ELSE
	  Statements
	ENDIF

ELSE

Syntax:	ELSE
Parameters:	None

Examples:	IF(@UserVar@ = 1)
	  @Count@ = @Count@ + 1
	ELSE
	  @UserVar@ = 0
	  Message("Tap OK to Continue","User Message","OK",""
	ENDIF

Notes:	See IF

ELSEIF

Syntax:	ELSEIF(<expression>)
Parameters:	
<expression>	Logical expression to be evaluated

Examples:	IF(@UserVar@ = 1)
	  @Count@ = @Count@ + 1
	ELSEIF (@UserVar@ = 0)
	  @Count@ = @Count@ - 1
	ENDIF

Notes:	See IF

ENDIF

Syntax:	ENDIF
Parameters:	None
Examples:	See IF
Notes:	See IF

FOR

Syntax:	FOR(<init>, <expression>, <update>)
Parameters:	
< init >	Statement to initialize the FOR loop
< expression >	If the expression evaluates to TRUE, the statements in the FOR loop execute. If the expression evaluates to FALSE, the FOR loop is done
< update >	The statement that increments the looping variable, the update executes after the statements in the FOR loop execute for its current pass through the loop.
Examples:	FOR(@Var@=1,@Var@<10,@Var@=@Var@+1) Message("Next Count: " & @Count@,"User Message","OK","") NEXT
Notes:	A FOR Loop. This can take the form: FOR(<init>, <expression>, <update>) Statements NEXT

NEXT

Syntax:	NEXT
Parameters:	None
Examples:	See FOR
Notes:	See FOR

EXITFOR

Syntax:	EXITFOR
Parameters:	None

Examples:	FOR(@Var@=1,@Var@<10,@Var@=@Var@+1)
	  @Msg@ = Message("Count: " & @Var@,"User Message","OK","Stop")
	  IF (@Msg@ = 2)
	  EXITFOR
	  ENDIF
	NEXT

Notes:	Exits a FOR Loop

WHILE

Syntax:	WHILE(<expression>)
Parameters:	
<expression>	Logical expression to be evaluated

Examples:	@Count@ = 0
	WHILE(@Count@ < 10)
	  @Count@ = @Count@ + 1
	  Message("Count: " & @Count@,"User Message","OK","")
	WEND

Notes:	A WHILE Loop. This can take the form: WHILE(<expression>) Statements WEND

WEND

Syntax:	WEND
Parameters:	None
Examples:	See WHILE
Notes:	See WHILE

EXITWHILE

Syntax:	EXITWHILE
Parameters:	None

Examples:	WHILE(TRUE)
	  IF (@Count@ => 10)
	  EXITWHILE
	  ENDIF
	  @Count@ = @Count@ + 1
	WEND

Notes:	Exits a WHILE loop

EXIT

Syntax:	EXIT
Parameters:	None

Examples:	FOR(@Var@=1,@Var@<10,@Var@=@Var@+1)
	  @Msg@ = Message("Count: " & @Var@,"User Message","OK","Stop")
	  IF (@Msg@ = 2)
	  EXIT
	  ENDIF
	NEXT

Notes:	Exits a Script

Constants

Input Sources

srcKeyboard	K
srcScanner	S
srcImage	I
srcText	T
srcEither	E

Barcode Symbologies

bcAny	0	bcMicroPDF417	23
bcCode39	1	bcPDF417	24
bcI2of5	2	bcOCR	25
bcCode128	3	bcPostnet	26
bcUPCA	4	bcJapanesePostal	27
bcCode11	5	bcDutchPostal	28
bcCode93	6	bcCanadianPostal	29
bcIATA	7	bcBritishPostal	30
bcCodabar	8	bcAustralianPostal	31
bcMSIPlessey	9	bcTriOptic	32
bcCodablock	10	bcChinaPostal	33
bcCode49	11	bcCode16K	34
bcECC.UCC	12	bcCode32	35
bcEAN	13	bcCouponCode	36
bcISBT	14	bcKoreaPostal	37
bcMSI	15	bcMatrix2of5	38
bcPlanetCode	16	bcPosicode	39
bcPlessey	17	bcStraight2of5	40
bcQRCode	18	bcTLCode39	41
bcRSS	19	bcTelepen	42
bcAztec	20	bcUCCEAN128	43
bcDataMatrix	21	bcUPCE	44
bcMaxiCode	22	bcEAN8	45

Logical

TRUE	1
FALSE	0

Validation File Modes

valLookup	0
valFound	1
valNotFound	2
valPickList	3

Keyboard Modes

keyNoChange	0
keyNumericMode	1
keyUppercaseAlpha	2
keyLowercaseAlpha	3

ASCII Values

ascCR	13
ascLF	10
ascTAB	9
ascESC	27

Date Formats

dateYear	Y
dateMonth	M
dateDay	D
dateHour	H
dateMinute	N
dateSecond	S
dateMMDDYYYY	M
dateMMDDYY	S
dateDDMMYYYY	D
dateDDMMYY	E
dateHHMMSS	T
dateHHMMSSAM	A
dateHHMM	H
dateHHMMAM	P

Serial Port

serCOM1	1
serCOM2	2
serCOM3	3
serCOM4	4
serCOM5	5
serCOM6	6
serCOM7	7
serCOM8	8
serCOM9	9
serBaud1200	1200
serBaud2400	2400
serBaud4800	4800
serBaud9600	9600
serBaud19200	19200
serBaud38400	38400
serBaud57600	57600
serData8	8
serData7	7
serParityNone	N
serParityEven	E
serParityOdd	O

Image Capture / Digital Ink Captions

capNone	0
capUpperLeft	1
capUpperCenter	2
capUpperRight	3
capLowerLeft	4
capLowerCenter	5
capLowerRight	6
capNoDate	0
capDateTime	1
capDate	2
capTime	3

Colors

colorBlack	0	RGB(0, 0, 0)
colorBlue	16711680	RGB(0, 0, 255)
colorBrown	16512	RGB(128, 64, 0)
colorCyan	16776960	RGB(0, 255, 255)
colorDarkBlue	8388608	RGB(0, 0, 128)
colorDarkGray	4210752	RGB(64, 64, 64)
colorDarkGreen	32768	RGB(0, 128, 0)
colorGray	8421504	RGB(128, 128, 128)
colorGreen	65280	RGB(0, 255, 0)
colorLtGray	12632256	RGB(192, 192, 192)
colorMagenta	16711935	RGB(255, 0, 255)
colorMaroon	128	RGB(128, 0, 0)
colorOrange	33023	RGB(255, 128, 0)
colorPurple	4194432	RGB(128, 0, 64)
colorRed	255	RGB(255, 0, 0)
colorWhite	16777215	RGB(255, 255, 255)
colorYellow	65535	RGB(255, 255, 0)

Event Options

eventDoNothing	0
eventNextElement	1
eventAcceptPrompt	2

Button Actions

buttonScriptOnly	0
buttonAcceptPrompt	1
buttonExitPrompt	2

OMNI Modes

omniManual	0
omniRF	1
omniHybridQuiet	2
omniHybridRetry	3
omniFailQuiet	1
omniRetryContinue	2

Radio Modes

radioNone	0
radioBT	1
radioWLAN	2
radioWLAN_BT	3
radioGSM	4
radioGSM_BT	5
radioWLAN_GSM	6
radioWLAN_GSM_BT	7
radioUnknown	-1

System

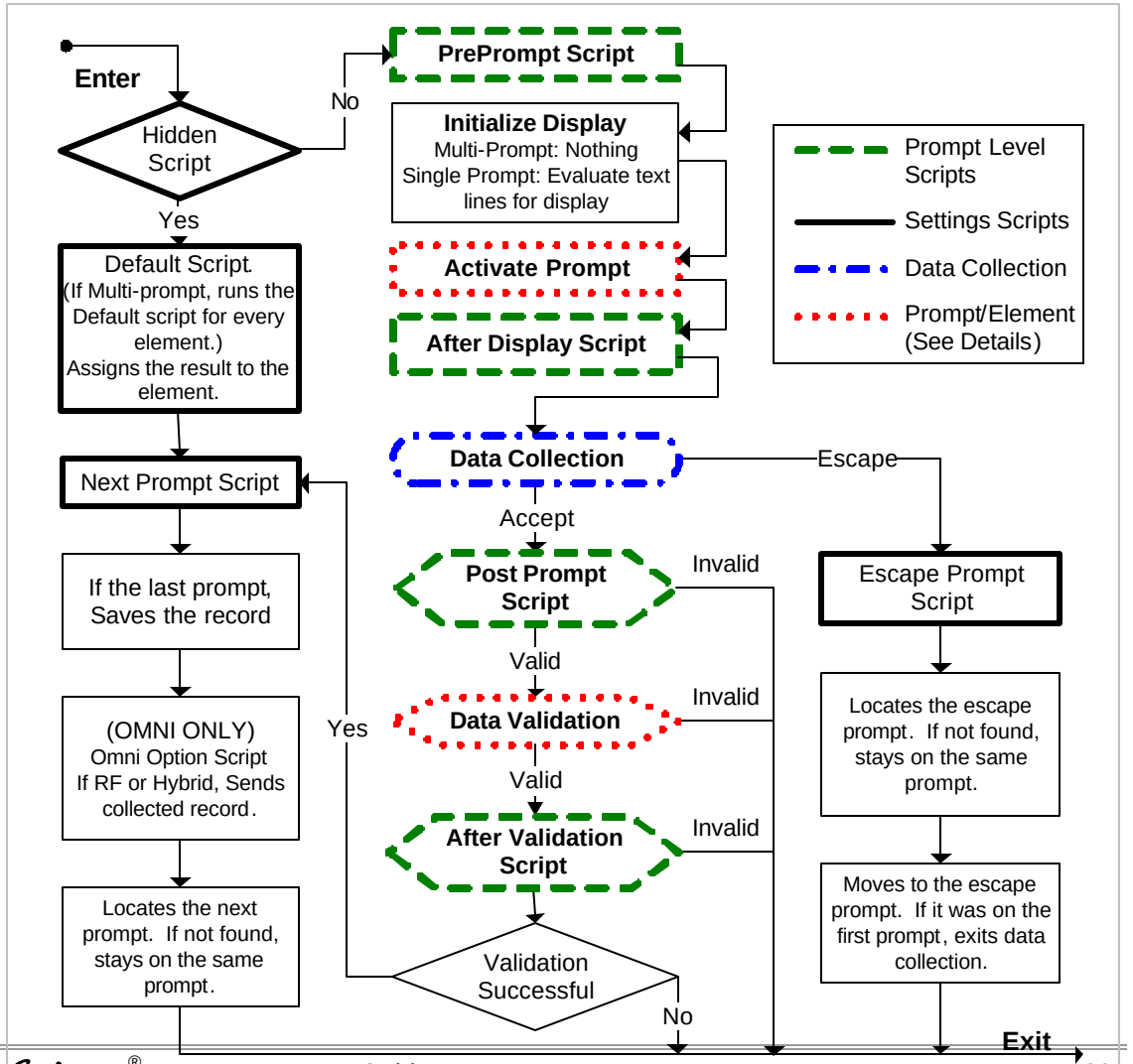
sysITBFile	The filename of the currently running ITB program.
sysITCFile	The filename of the current collected data file.
sysTerminalID	The numeric identifier of the client. See the client-specific documentation for the Terminals IDs.
sysScreenWidth	The total width of the device screen in pixels.
sysScreenHeight	The total height of the device screen in pixels.
sysClientWidth	The usable client width of the program in pixels.
sysClientHeight	The usable client height of the program in pixels.

Script Sequencing Reference

This section documents the order in which scripts are executed under various conditions.

Prompt Lifecycle

This diagram shows the basic lifecycle of a prompt. As a user enters data and moves from prompt to prompt in the data collection program, each prompt proceeds through a series of steps to activate the prompt, run the various In-Prompt scripts, collect data, and evaluate which prompt to



go to next. Most of the time, the details of the exact order of these steps and scripts within the life-cycle of a prompt is not important. However, there may times when the interaction between In-Prompt scripts is such that the exact order of execution needs to be clearly understood. The diagram is applicable to both Single-Prompts and Multi-Prompts. This section will clarify the order of Prompt and/or element scripts for both prompt types.

Single Prompt

The life cycle of a Single-Prompt starts with a check on whether or not the prompt is a Hidden prompt. The Hidden setting is one of the settings on the *Advanced Settings* screen for a Single Prompt. If there is an In-Prompt Script that determines if the prompt is hidden, it is executed to find out if the prompt will be hidden or not. If the Prompt is hidden, only the Default script and the Next Prompt script will run before the prompt exits and a new prompt is begun. In most cases, the prompt will not be hidden and the prompt will execute fully as shown in the diagram. The Pre-Prompt Script (which is accessed from the *Before Prompting* button on the *Advanced Settings* screen) is then run followed by **IT*Script*Net** calculating the display text to initialize the display.

When a Single Prompt is displayed, the Activation scripts are run. These are In-Prompt scripts to determine many of the settings for a prompt. The order of the In-Prompt scripts for activation is:

In-Prompt Scripts for Activation
Max Length
Min Length
Input Source
Symbology
Allow Blank
First Character
Reverse
Default To Last
Default Value
Mask
Keyboard Mode
Centering/ALD
Validation Type
If the prompt is a Picklist, the following scripts are also run:
Validation File
Validation Field
Lookup Field
Filter Field
Filter Value
Format Specifier
Picklist Script (once per record as added to the list).

Once the prompt has been activated and displayed, the After Display script (defined in the Advanced Settings screen) is run. The user now is able to enter or scan data as a response to the prompt.

After the data is entered, the Post-Prompt script is executed (This is the script behind the After Prompting button on the Advanced Settings screen). **IT*Script*Net** then handles the built-in validations on the data that was entered. The first step towards data validation is to run any of the settings scripts to establish what various data validation settings values will be.

In-Prompt Scripts for Data Validation
Store First Char
Store Mask
Range Checking
Range Hi Value
Range Lo Value
Validation File
Validation Type
Validation Field
Lookup Field

The entered data can now be checked against the built-In Validations in the following order:

Built-In Validations
Allow Blank
Min Length
Max Length
First Character
Mask
Range
Validation (Must Be found, etc).

After **IT*Script*Net** runs the built-in validation checks on the data, the After Validation In-Prompt script is run before final determination is made as to whether the data entered has passed. If the data is determined to be valid, the Next Prompt In-Prompt script is run and the program proceeds to the next prompt.

Multi-Prompt

The life cycle of a Multi-Prompt starts with a check on whether or not the prompt is a Hidden prompt. The *Multi-Prompt Settings* screen contains an option to "Do not prompt user (Skip)". If this setting is checked or if the In-Prompt script for this setting evaluates to True, the prompt is hidden. If the Prompt is hidden, only the Default script and the Next Prompt script will run before the prompt exits and a new prompt is begun. In most cases, the prompt will not be hidden and the prompt will execute fully as shown in the diagram. The Pre-Prompt Script (which is accessed from the *Before Prompting* button on the *Multi-Prompt Settings* screen) is then run. For a Multi-Prompt the Activation step is next and includes performing the tasks shown in the table below.

In-Prompt Scripts for Activation
Prompt-level: Use Custom Colors
Prompt-level: Custom Color
Run Activation Scripts for each Display Element
Run Activation Scripts for each Input Element

Note that there are only a few prompt-level aspects to the activation step. The majority of the activation is running the activation scripts for each of the elements on the prompt. In the sections that follow, each type of element is included showing a table of In-Prompt scripts that execute on activation.

Once the prompt has been activated and displayed, the After Display script (defined in the *Multi-Prompt Settings* screen) is run. The user now is able to enter or scan data as a response to the input elements on the prompt. The user may tab or click to move the input focus from one element to the other. As each element loses the focus, gets, the focus, or is clicked, that element's *On Lose Focus*, *On Get Focus*, or *On Clicked* Event In-Prompts run accordingly. During data entry, if the Refresh function is run on an element, the Activation scripts for that element are run again (the same activations that occur when the element is activated prior to being displayed.

At some point in data collection, the user will click on a button that is set to trigger the program to move on to the next prompt. Depending on the design of the prompt, scanning or tapping the enter button might also trigger the program to move on to the next prompt. Once the trigger to go the Next Prompt has occurred, **IT*Script*Net** will continue the prompt's life-cycle as shown in the diagram. The Post-Prompt script is executed (this is the script behind the After Prompting button on the *Multi-Prompt Settings* screen) followed by the Data Validation Step. The Data Validation step involves running Validation In-Prompt scripts and the Validation event for each input element. See the following pages for the details of what scripts are run for each element type during Validation. After **IT*Script*Net** runs the built-in validation checks on the data, the After Validation In-Prompt script is run before final determination is made as to whether the data entered has passed. If the data is determined to be valid for all the input elements on the prompt, the Next Prompt In-Prompt script is run and the program proceeds to the next prompt.

Elements

Display Elements: Text, Textlist, Image, Shape

Text	Text List	Image	Shape
Activate	***Activate***	***Activate***	***Activate***
Display Text	Use Custom Colors	Filename Script	Left
Translates $_$, @_@, #_# variables	Foreground Color	Left	Top
	Background Color	Top	Height
Left	Disabled Foreground Color	Height	Width
Top		Width	Hidden
Height	Disabled Background Color	Hidden	Border Color
Width			Border Width
Foreground Color	Left		Fill Color
Background Color	Top		Transparent
Transparent	Height		
Shadow Color	Width		
Use Shadow	Font Name		
Shadow Distance	Font Height		
Font Name	Font Width		
Font Height	Bold		
Font Width	Italic		
Hidden	Strikeout		
Bold	Underline		
Italic	Text		
Strikeout	Hidden		
Underline			
Justification			

Input Text

Activate	*Validation Scripts*
Use Custom Colors	Allow Blank
Foreground Color	First Character
Background Color	Store First
Disabled Foreground Color	Mask
Disabled Background Color	Store Mask
Left	Range Checking
Top	Range Hi
Height	Range Lo
Width	Val File
Font Name	Val Type
Font Height	Val Field
Font Width	Lookup Field
Bold	Validation Event
Italic	
Strikeout Underline	***Get Focus*** When the element receives the focus, the following scripts are run:
Max Length	Max Len
Min Length	Min Len
Default To Last	Input Source
Default Value	Symbology
Disabled	Default To Last
Hidden	Default: The default value is changed only if the edit box is blank
First Character	Keyboard Mode
Store First Character	
Mask Store Mask	***After Scan*** When a barcode is scanned, the following script is run: AfterScan
Range Check	
Range Hi Range Lo	***After Enter*** When Enter is pressed, the following script is run: After Enter
Validation Type	
Validation File	
Validation Field	
Lookup Field	
Input Source	
Symbology	
Centering	
ALD	
Allow Blank	
Follow Mask	
Message Override	
Password	
Force Case	

Action Button, Checkbox, Radio Button

Button	Checkbox	Radio Button
Activate	***Activate***	***Activate***
Use Custom Colors	Use Custom Colors	Use Custom Colors
Foreground Color	Foreground Color	Foreground Color
Background Color	Background Color	Background Color
Disabled Foreground Color	Disabled Foreground Color	Disabled Foreground Color
Disabled Background Color	Disabled Background Color	Disabled Background Color
Left	Left	Font Name
Top	Top	Font Height
Height	Height	Font Width
Width	Width	Bold
Font Name	Font Name	Italic
Font Height	Font Height	Strikeout
Font Width	Font Width	Underline
Bold	Bold	Default To Last
Italic	Italic	Default Value
Strikeout	Strikeout	Allow Blank
Underline	Underline	Disabled
Disabled	Disabled	Hidden
Hidden	Hidden	Input Source
Text	Text	Symbology
Action	Selected Value	Centering
	Unselected Value	ALD
	Default Value	
Validation Scripts	***Validation Scripts***	***Validation Scripts***
There are no validation scripts for this element	There are no validation scripts for this element	Allow Blank
Clicked Script	***Clicked Script***	***Clicked Script***
The Clicked Script is run before the Accept or Escape action occurs.	The check state is changed before the Clicked Script is run	The check state is changed before the Clicked Script is run.
		Get Focus
		When the element receives the focus, the following scripts are run:
		Default To Last
		Default Value

ComboBox, ListBox, and Grid

Activate	***Validation Scripts***
Left	Allow Blank
Top	
Height	***Clicked Script***
Width	The Clicked Script is run each time the selected item is changed.
If Dynamic Content, runs the following:	
Validation File	Default To Last
Validation Field	Default Value
Lookup Field	Note: The selected value is changed only if no item is currently selected.
Filter Field	
Filter Value	
Format Specifier	***After Scan Script***
As the list is filled, PickList Script is called once per record.	Run when a barcode is scanned.
Use Custom Colors	
Foreground Color	***After Enter Script***
Background Color	Run when Enter is pressed.
Disabled Foreground Color	
Disabled Background Color	
Font Name	
Font Height	
Font Width	
Bold	
Italic	
Strikeout	
Underline	
Default To Last	
Default Value	
Allow Blank	
Disabled	
Hidden	
Input Source	
Symbology	
Centering	
ALD	

MultiList Element

Activate
Use Custom Colors
Foreground Color
Background Color
Disabled Foreground Color
Disabled Background Color
Left
Top
Height
Width
Font Name
Font Height
Font Width
Bold
Italic
Strikeout
Underline
Header Text
Default To Last
Allow Blank
Disabled
Hidden
As each item is added to the list, the following item scripts are run:
Text
Selected Value
Unselected Value
Default
Validation Scripts
Allow Blank
Get Focus
When the element receives the focus, the following scripts are run:
Header Text
Default To Last

Image Capture and Digital Ink

Image Capture	Digital Ink
Activate	***Activate***
Use Custom Colors	Use Custom Colors
Foreground Color	Foreground Color
Background Color	Background Color
Disabled Foreground Color	Disabled Foreground Color
Disabled Background Color	Disabled Background Color
Left	Left
Top	Top
Height	Height
Width	After Enter
Font Name	Disabled
Font Height	Hidden
Font Width	Allow Blank
Bold	
Italic	
Strikeout	
Underline	
Disabled	
Hidden	
Text	
Image Profile	
Caption Text	
Caption Font Height	
Caption Black	***Validation Scripts***
Caption Transparent	Caption Text
Datestamp Overlay	Caption Font Height
Datestamp Font Height	Caption Black
Datestamp Black	Caption Transparent
Datestamp Transparent	Datestamp Overlay
Allow Blank	Datestamp Font Height
Validation Scripts	Datestamp Black
None	Datestamp Transparent
Clicked Script	***Clicked Script***
The Clicked Script is run before the Image Capture action occurs.	The Clicked Script is run when user clicks in digital ink area

Multiprompt Functions

Not all of the Multiprompt scripts are effective for every element type. For example, the Select function has no effect on a Display Text element, but it does on an Input Text element. The following charts describe how each multiprompt function behaves for each element type.

Clear

Element	Applies?	Note
Display Text		
Image		
Shape		
TextList	Y	Clears any text from the TextList.
Timer	Y	Disables the Timer.
Button		
CheckBox		
ComboBox	Y	Clears all items from the ComboBox.
DigitalInk	Y	Erases any image data and clears the control.
Grid	Y	Clears all items from the Grid.
ImageCapture		
ListBox	Y	Clears all items from the ListBox.
MultiList	Y	Clears all items from the MultiList.
RadioButton		
TextBox	Y	Clears any text from the TextBox.

Disable

For most elements, Disable causes the element to stop responding to keypresses or clicks, and to not accept the focus.

Element	Applies?	Note
Display Text		
Image		
Shape		
TextList	Y	
Timer	Y	Stops the timer.
Button	Y	
CheckBox	Y	
ComboBox	Y	
DigitalInk	Y	
Grid	Y	
ImageCapture	Y	
ListBox	Y	
MultiList	Y	
RadioButton	Y	
TextBox	Y	

Enable

Element	Applies?	Note
Display Text		
Image		

Shape		
TextList	Y	
Timer	Y	Restarts the time interval.
Button	Y	
CheckBox	Y	
ComboBox	Y	
DigitalInk	Y	
Grid	Y	
ImageCapture	Y	
ListBox	Y	
MultiList	Y	
RadioButton	Y	
TextBox	Y	

Hide

Element	**Applies?**	**Note**
Display Text	Y	
Image	Y	
Shape	Y	
TextList	Y	
Timer		
Button	Y	
CheckBox	Y	
ComboBox	Y	
DigitalInk	Y	
Grid	Y	
ImageCapture	Y	
ListBox	Y	
MultiList	Y	
RadioButton	Y	Hides the entire group of radio buttons.
TextBox	Y	

IsEnabled

Returns 1 if the element is enabled, or 0 if it is not.

Element	Applies?	Note
Display Text		Always returns 0
Image		Always returns 0
Shape		Always returns 0
TextList	Y	
Timer	Y	
Button	Y	
CheckBox	Y	
ComboBox	Y	
DigitalInk	Y	
Grid	Y	
ImageCapture	Y	
ListBox	Y	
MultiList	Y	
RadioButton	Y	
TextBox	Y	

IsVisible

Returns 1 if the element is visible, or 0 if it is not.

Element	Applies?	Note
Display Text	Y	
Image	Y	
Shape	Y	
TextList	Y	
Timer		Always returns 0
Button	Y	
CheckBox	Y	
ComboBox	Y	
DigitalInk	Y	
Grid	Y	
ImageCapture	Y	
ListBox	Y	
MultiList	Y	
RadioButton	Y	
TextBox	Y	

Refresh

For most elements, Refresh causes the Activate scripts to run.

Element	Applies?	Note
DisplayText	Y	Runs Activate scripts.
Image	Y	Runs Activate scripts.
Shape	Y	Runs Activate scripts.
TextList	Y	Runs Activate scripts.
Timer	Y	Runs Activate scripts.
Button	Y	Runs Activate scripts.
CheckBox	Y	Runs Activate scripts.
ComboBox	Y	Runs Activate scripts and re-fills the ComboBox.
DigitalInk	Y	Runs Activate scripts.
Grid	Y	Runs Activate scripts and re-fills the Grid.
ImageCapture	Y	Runs Activate scripts.
ListBox	Y	Runs Activate scripts and re-fills the ListBox.
MultiList	Y	Runs Activate scripts and re-fills the MultiList.
RadioButton	Y	Runs Activate scripts.
TextBox	Y	Runs Activate scripts.

Select

Element	Applies?	Note
DisplayText		
Image		
Shape		
TextList		
Timer		
Button	Y	Simulates a button press. Also runs the Clicked script.
CheckBox		
ComboBox	Y	Sets the current selection of the Combobox to the item number specified. The first element is 1.
DigitalInk		
Grid	Y	Sets the current selection of the grid to the item number specified. The first element is 1.
ImageCapture	Y	Simulates a button press. Also runs the Clicked script.
ListBox	Y	Simulates a button press. Also runs the Clicked script.
MultiList	Y	Sets the current selection of the MultiList to the item number specified. The first element is 1.
RadioButton		
TextBox	Y	Selects or unselects the text. If 1, selects the text in the TextBox. If 0, unselects the text.

SetFocus

Element	Applies?	Note
Display Text		
Image		
Shape		
TextList		
Timer		
Button	Y	
CheckBox	Y	
ComboBox	Y	
DigitalInk	Y	
Grid	Y	
ImageCapture	Y	
ListBox	Y	
MultiList	Y	
RadioButton	Y	Sets the focus to the currently select button in the group.
TextBox	Y	

Show

Element	Applies?	Note
Display Text	Y	
Image	Y	
Shape	Y	
TextList	Y	
Timer		
Button	Y	
CheckBox	Y	
ComboBox	Y	
DigitalInk	Y	
Grid	Y	
ImageCapture	Y	
ListBox	Y	
MultiList	Y	
RadioButton	Y	Shows the entire group of radio buttons.
TextBox	Y	

This page intentionally left blank