



## Instruction Manual

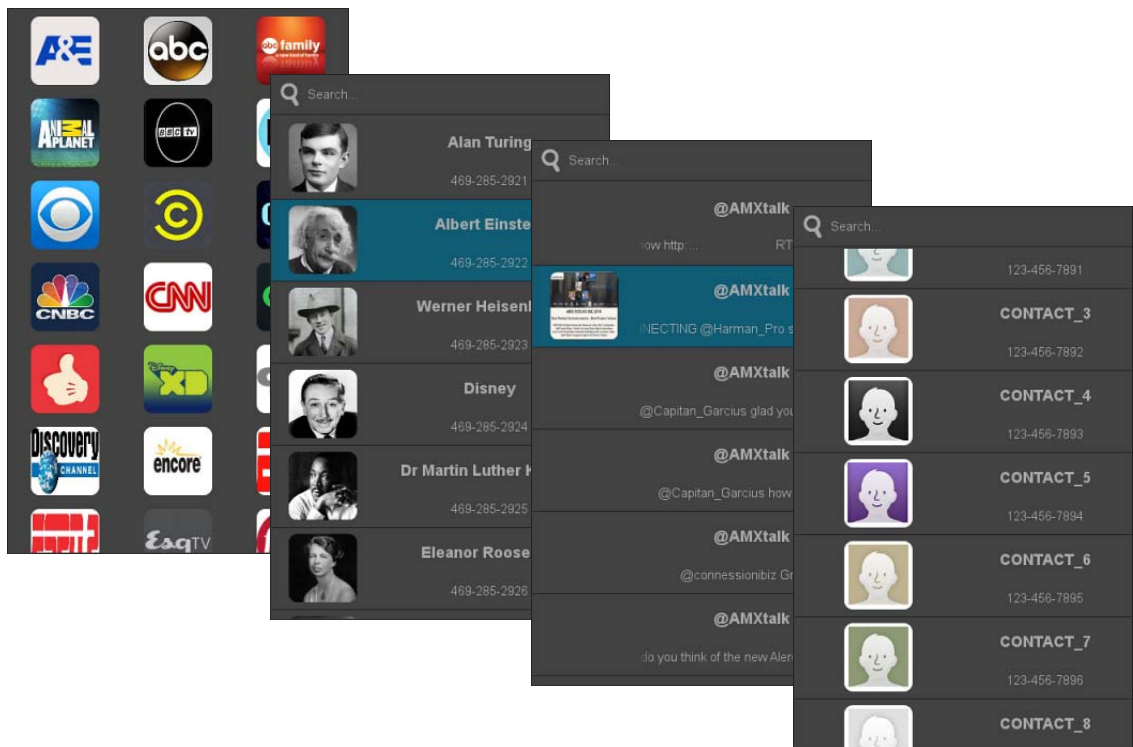
# Listview Buttons & Dynamic Data Demo Files

Example 1: CSV File - With Headers

Example 2: CSV File - No Headers

Example 3: XPort Server-Generated XML

Example 4: NetLinx Data Source



# AMX Limited Warranty and Disclaimer

This Limited Warranty and Disclaimer extends only to products purchased directly from AMX or an AMX Authorized Partner which include AMX Dealers, Distributors, VIP's or other AMX authorized entity.

AMX warrants its products to be free of defects in material and workmanship under normal use for three (3) years from the date of purchase, with the following exceptions:

- Electroluminescent and LCD Control Panels are warranted for three (3) years, except for the display and touch overlay components are warranted for a period of one (1) year.
- Disk drive mechanisms, pan/tilt heads, power supplies, and MX Series products are warranted for a period of one (1) year.
- AMX lighting products are guaranteed to switch on and off any load that is properly connected to our lighting products, as long as the AMX lighting products are under warranty. AMX also guarantees the control of dimmable loads that are properly connected to our lighting products. The dimming performance or quality there of is not guaranteed, impart due to the random combinations of dimmers, lamps and ballasts or transformers.
- AMX software is warranted for a period of ninety (90) days.
- Batteries and incandescent lamps are not covered under the warranty.
- AMX AutoPatch Epica, Modula, Modula Series4, Modula CatPro Series and 8Y-3000 product models will be free of defects in materials and manufacture at the time of sale and will remain in good working order for a period of three (3) years following the date of the original sales invoice from AMX. The three-year warranty period will be extended to the life of the product (Limited Lifetime Warranty) if the warranty card is filled out by the dealer and/or end user and returned to AMX so that AMX receives it within thirty (30) days of the installation of equipment but no later than six (6) months from original AMX sales invoice date. The life of the product extends until five (5) years after AMX ceases manufacturing the product model. The Limited Lifetime Warranty applies to products in their original installation only. If a product is moved to a different installation, the Limited Lifetime Warranty will no longer apply, and the product warranty will instead be the three (3) year Limited Warranty.

All products returned to AMX require a Return Material Authorization (RMA) number. The RMA number is obtained from the AMX RMA Department. The RMA number must be clearly marked on the outside of each box. The RMA is valid for a 30-day period. After the 30-day period the RMA will be cancelled. Any shipments received not consistent with the RMA, or after the RMA is cancelled, will be refused. AMX is not responsible for products returned without a valid RMA number.

AMX is not liable for any damages caused by its products or for the failure of its products to perform. This includes any lost profits, lost savings, incidental damages, or consequential damages. AMX is not liable for any claim made by a third party or by an AMX Authorized Partner for a third party.

This Limited Warranty does not apply to (a) any AMX product that has been modified, altered or repaired by an unauthorized agent or improperly transported, stored, installed, used, or maintained; (b) damage caused by acts of nature, including flood, erosion, or earthquake; (c) damage caused by a sustained low or high voltage situation or by a low or high voltage disturbance, including brownouts, sags, spikes, or power outages; or (d) damage caused by war, vandalism, theft, depletion, or obsolescence.

This limitation of liability applies whether damages are sought, or a claim is made, under this warranty or as a tort claim (including negligence and strict product liability), a contract claim, or any other claim. This limitation of liability cannot be waived or amended by any person. This limitation of liability will be effective even if AMX or an authorized representative of AMX has been advised of the possibility of any such damages. This limitation of liability, however, will not apply to claims for personal injury.

Some states do not allow a limitation of how long an implied warranty last. Some states do not allow the limitation or exclusion of incidental or consequential damages for consumer products. In such states, the limitation or exclusion of the Limited Warranty may not apply. This Limited Warranty gives the owner specific legal rights. The owner may also have other rights that vary from state to state. The owner is advised to consult applicable state laws for full determination of rights.

EXCEPT AS EXPRESSLY SET FORTH IN THIS WARRANTY, AMX MAKES NO OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. AMX EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED IN THIS LIMITED WARRANTY. ANY IMPLIED WARRANTIES THAT MAY BE IMPOSED BY LAW ARE LIMITED TO THE TERMS OF THIS LIMITED WARRANTY. EXCEPT AS OTHERWISE LIMITED BY APPLICABLE LAW, AMX RESERVES THE RIGHT TO MODIFY OR DISCONTINUE DESIGNS, SPECIFICATIONS, WARRANTIES, PRICES, AND POLICIES WITHOUT NOTICE.

# Table of Contents

<b>Listview Buttons &amp; Dynamic Data .....</b>	<b>1</b>
Overview .....	1
AMX System Requirements for Listview Buttons .....	1
Updating the NetLinx.AXI File to v1.55 .....	2
Determining the Current Version of the NetLinx.AXI File .....	2
Updating the NetLinx.AXI File .....	2
Working With Listview Button Properties.....	3
Listview Buttons - General Properties .....	3
Listview Buttons - Programming Properties .....	3
Listview Buttons - States Properties .....	3
Listview Buttons - Events Properties .....	3
<b>Example 1: CSV File - With Headers .....</b>	<b>5</b>
Overview .....	5
Before You Begin .....	5
1) Create (draw) a Listview button .....	5
2) Set the Listview Button Properties.....	6
3) Host the Data Source File (CSV with Headers) on the NX Master .....	7
4) Add the Dynamic Data Source to the Project.....	9
5) Map the Data from the Data Source File to the Listview Button Components .....	10
Step One: Analyze the Data Source.....	10
Step Two: Map the Data to Components of the Listview button .....	11
Dynamic Data Mappings - Syntax Requirements (CSV with Headers) .....	12
6) Add Image Files to the Project.....	12
7) Assign a Data Source file to the Listview Button.....	14
8) Write a Custom Event To Respond To User Selection .....	15
9) Transfer the TPDesign5 Project to the Touch Panel .....	17
Example 1 (CSV File - With Headers) - Results .....	18
Reference: "channelList.csv" (CSV File With Headers) .....	19
TV Guide Demo File ("TVGuide.ZIP") .....	20
<b>Example 2: CSV File - No Headers .....</b>	<b>23</b>
Overview .....	23
Before You Begin .....	23
1) Create (draw) a Listview button .....	23
2) Review the Listview Button Properties.....	24
3) Host a Data Source File (CSV without Headers) on the NX Master .....	25
4) Add the Dynamic Data Source to the Project.....	27
5) Map the Data from the Data Source File to the Listview Button Components .....	28
Step One: Analyze the Data Source.....	28
Step Two: Map the Data to Components of the Listview button .....	29

Dynamic Data Mappings - Syntax Requirements (CSV Without Headers) .....	30
6) Add Image Files to the Project.....	30
7) Assign a Data Source file to the Listview Button.....	32
8) Write a Custom Event To Respond To User Selection .....	33
9) Transfer the TPDesign5 Project to the Touch Panel .....	35
Example 2 (CSV File - No Headers) - Results .....	36
Reference: "conference.csv" (CSV File Without Headers).....	37
Conference Rooms Demo File ("Conference.ZIP").....	37
<b>Example 3: XML File/XPort Server .....</b>	<b>39</b>
Overview .....	39
Before You Begin .....	39
1) Create Twitter Feed on the XPort Server.....	39
2) Generate the "amxstandard.xml" file.....	42
3) Create (draw) a Listview button .....	42
4) Set Listview Button Properties .....	43
5) Add Dynamic Data Source to the Project.....	43
6) Map the Data from the Data Source File to the Listview Button Components .....	45
Step One: Analyze the Data Source.....	45
Step Two: Map the Data to Components of the Listview button .....	45
Dynamic Data Mappings - Syntax Requirements (XPort-Generated XML) .....	47
7) Assign a Data Source file to the Listview Button.....	48
8) Write a Custom Event To Respond To User Selection .....	48
9) Transfer the TPDesign5 Project to the Touch Panel .....	50
Example 3 (XML File/XPort Server) - Results.....	51
Reference: "amxstandard.xml" .....	52
Twitter (XPort XML) Demo File ("Twitter.ZIP") .....	52
<b>Example 4: NetLinX Data Source .....</b>	<b>55</b>
Overview .....	55
Before You Begin .....	55
1) Create (draw) a Listview button .....	55
2) Set Listview Button Properties .....	56
3) Create the Data Source .....	56
4) Configuring the Response to a User Selection .....	56
NetLinX Usage Example - ASCII .....	56
5) Compile the Code .....	60
6) Transfer the Workspace to the NX Master.....	60
Example 4 (NetLinX Data Source) - Results.....	61
NetLinXAPI Demo File ("NetLinXAPI.ZIP") .....	62
Listview (Data Access) Send Commands .....	63
Terminology .....	63
DataFeed .....	63
DataRecord .....	63
DataField.....	63

^LVC .....	63
^LVD .....	64
^LVF.....	64
^LVL.....	65
^LVM.....	67
^LVN .....	67
^LVR.....	68
^LVS.....	69
<b>Listview Button Properties .....</b>	<b>71</b>
<b>Overview .....</b>	<b>71</b>
<b>General Properties:.....</b>	<b>71</b>
Alphabet Scrollbar .....	71
Dynamic Data Source .....	71
Filter Enabled.....	71
Filter Height .....	71
Item Height .....	72
Listview Columns.....	72
Listview Components .....	72
Listview Item Layout .....	74
Primary Partition (%) .....	75
Secondary Partition (%).....	76
<b>Programming Properties.....</b>	<b>77</b>
Address Port .....	77
Address Code.....	77
<b>State Properties.....</b>	<b>78</b>
Secondary Font .....	78
Secondary Font Size.....	78



# Listview Buttons & Dynamic Data

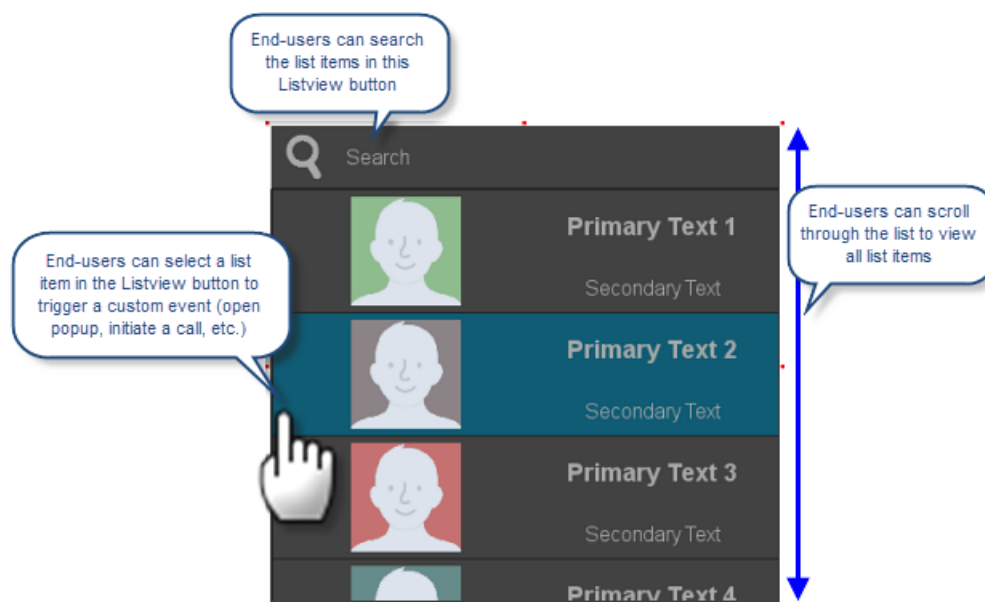
## Overview

Modero X Series G5 touch panels and TPDesign5 (v1.2.0, build 65 or greater) support *Listview* buttons. Listview buttons provide the ability to display a listing of items from a dynamic data source on a G5 touch panel. Dynamic data can be created either using an XPort server, NetLinx code or a generic CSV file. The creator of the data can specify how many fields comprise a record and the format of those fields. As many records as necessary can be specified.



*Dynamic data defines data files/feeds URL where the data can be loaded by the touch panel at runtime via HTTP (GET) or HTTPS (GET) transport protocols.*

This data can be used to populate a Listview button displayed on a G5 touch panel, where the end user can scroll or search through the list and make a selection. Once a selection has been made, a CUSTOM\_EVENT is raised in the NX Master to retrieve the data fields comprising the selected record. An example Listview button is shown in FIG. 1:



**FIG. 1** Example Listview button (in TPDesign5)

End users can scroll through the items in the list, and select an item to initiate a custom event. Using the Outlook contacts list as an example, the end user could select a name in the Listview button to view contact information for the selected name, or call that contact directly from the panel, depending on NetLinx programming and settings in TPDesign5.

## AMX System Requirements for Listview Buttons

The following software, hardware and firmware requirements must be met to support Listview buttons:

- TPDesign5 - version 1.2.0 build 65 (or greater)
- X Series G5 Touch Panels - panel firmware version 1.3.10 (or greater)
- NetLinx NX Series Masters - master firmware version 1.3.17 (or greater)
- NetLinx.AXI file (version 1.55 or greater)



*To determine the version number of the NetLinx.AXI file currently loaded, refer to the NetLinx Studio 4 About dialog (**Help > About NetLinx Studio**). If your NetLinx.AXI file is older than version 1.55, use the **NetLinx Support File Update Setup** program to update the NetLinx.AXI file. See the *Updating the NetLinx.AXI File* to v1.55 section on page 2 for details.*

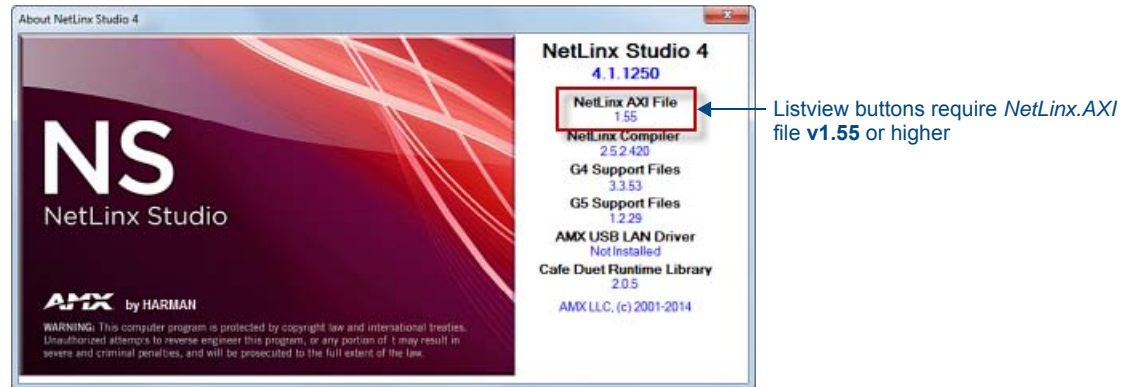
- A source for the data that will be presented in list form on the Listview button.

## Updating the NetLinX.AXI File to v1.55

In order to support Listview buttons and dynamic data, **NetLinX.AXI file v1.55 (or higher)** is required. If your NetLinX.AXI file is older than v1.55, use the *NetLinX Support File Update Setup* program to update the NetLinX.AXI file.

### Determining the Current Version of the NetLinX.AXI File

To determine the version number of the NetLinX.AXI file currently loaded, refer to the NetLinX Studio 4 *About* dialog (**Help > About NetLinX Studio**):



**FIG. 2** About NetLinX Studio 4 dialog, indicating NetLinX.AXI file version 1.55

### Updating the NetLinX.AXI File

1. Download the *NetLinX Support File Update Setup* file (**NetLinXSupportFileUpdateSetup.exe**) from [www.amx.com](http://www.amx.com) (FIG. 3):



**FIG. 3** NetLinX Support File Update installation program

2. Double-click the file to begin the installation, and click *Next* in each dialog to accept the default location for the updated files.
3. When the installation is complete, NetLinX Studio 4 is ready to support Listview buttons.



## Working With Listview Button Properties

While Listview button properties are the same as for other button types in many ways, it is important to understand that Listview buttons have several unique properties, and others that work differently for Listview buttons than for other button types:

### Listview Buttons - General Properties

The following General properties are specific to Listview buttons:

- Listview Components (see page 72)
- Item Height (see page 72)
- Listview Columns (see page 72)
- Listview Item Layout (see page 74)
- Primary Partition (%) (see page 75)
- Secondary Partition (%) (see page 76)
- Filter Enabled (see page 71)
- Filter Height (see page 71)
- Alphabet Scrollbar (see page 71)
- Dynamic Data Source (see page 71)

### Listview Buttons - Programming Properties

Once you have created a Listview button, you can use the Programming tab of the Properties window to set/edit programming-oriented button properties. To edit any of the properties, click in the right-hand table cell to activate the field. Depending on the item selected, you can either set the item manually, select from a drop-down menu, or both.

Listview buttons only use the Address Port and Address Code Programming properties:

- Address Port (see page 77)
- Address Code (see page 77)

Channel Port and Channel Code are not supported for Listview buttons.

### Listview Buttons - States Properties

Rather than the On/Off state options that apply to other button types, Listview buttons support the following two states:

- **Default** - The property values of the Default state will be used to render non-selected list items and also the button background, in the eventuality that there are not enough list items to fill the entirety of the Listview button.
- **Selected** - The property values of the Selected state will be used to render selected list items.



NOTE

*Note that in the TPD5 State Manager window, Default is labeled "Off" and Selected is labeled "On".*

Other than these new State types, the State properties supported by Listview buttons is basically the same as for other button types, with two properties that are specific to Listview buttons: *Secondary Font* and *Secondary Font Size*:

- Secondary Font (see page 78)
- Secondary Font Size (see page 78)

Note that the *Secondary Font* and *Secondary Font Size State* properties are available even if the selected Listview button only uses a single line of text. In this case, if the *List View Type* is changed to either two-line text or two-line text with icon, the second line of text will use these settings.

### Listview Buttons - Events Properties

Use the *Events* tab of the TPD5 Properties window to set event properties for the selected Listview button. Listview buttons support the following three Events properties that are specific to Listview buttons. However, these Events support the same *actions* as existing events.

- Item Selected (see page 232)
- Scrollbar Begin (see page 232)
- Scrollbar End (see page 232)



# Example 1: CSV File - With Headers

## Overview

The following instructions describe using the TV Guide demo for creating a Listview button with a dynamic data source in the form of a CSV file with headers that is hosted on an NX Master.



*This set of instructions uses files that are included in the "TV Guide.ZIP" demo file which is available to download from the UI RESOURCE CENTER at [www.amx.com](http://www.amx.com).*

The resulting Listview button will display a listing of TV channels with each channel's station icon in a three-column grid layout (FIG. 4):



FIG. 4 Example - Listview button based on "channelList.csv"

## Before You Begin

1. Download the *TVGuide.ZIP* file from the UI RESOURCE CENTER at [www.amx.com](http://www.amx.com) and extract it's contents to a known location.
2. Open the *channelList.csv* file and analyze it's contents. It is a relatively simple csv file that consists of four columns with headers (NAME, CHANNEL CODE, ICON and RATING). See page 19 to view this file.

### 1) Create (draw) a Listview button

1. In TPDesign5, open a Page and use the Button Draw tool to create a new button.
2. With the new button selected, click the **Type** (General) property and select **Listview** from the drop-down of button types. This selection sets the new button as a Listview button, and enables a set of Listview-specific properties (FIG. 5):

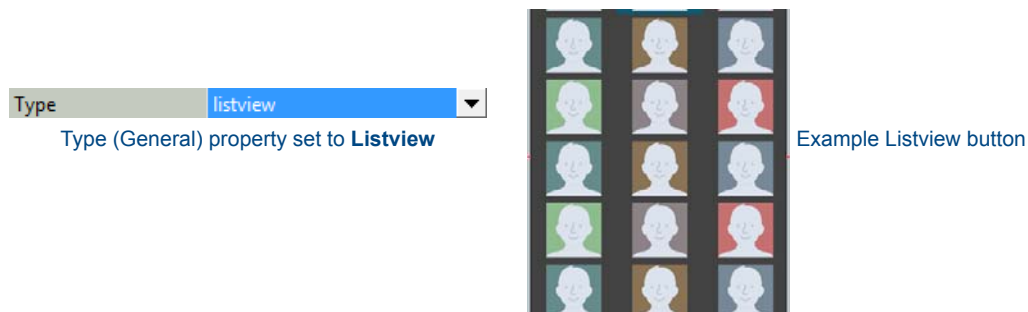


FIG. 5 Type (General) Property set to Listview



NOTE

The "TVGuide.TP5" file included in the TV Guide demo has a Listview button already drawn on the "Main" page.

## 2) Set the Listview Button Properties

Use the options in the Properties window to view/edit the *General*, *Programming* and *States* properties for the Listview button (this demo does not use *Events* properties). The settings used for the Listview button in the TV Guide demo are shown in FIG. 6:

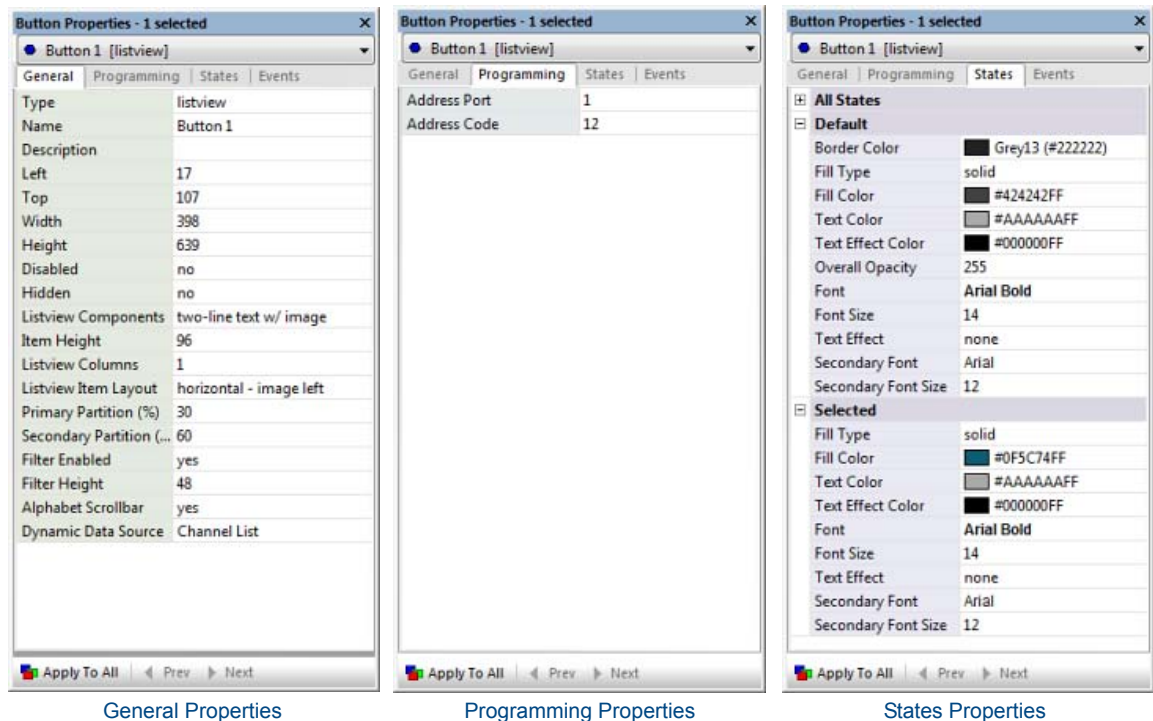


FIG. 6 Properties for the TV Guide Listview Button



NOTE

The Listview button in the TV Guide demo is pre-configured with the General, Programming and States properties shown above.

Refer to the *Working With Listview Button Properties* section on page 3 for details on Listview-specific button properties.

### 3) Host the Data Source File (CSV with Headers) on the NX Master

In this example, "channelList.csv" will be the data source for the Listview button. This CSV file will be hosted on the NX Master. "channelList.csv" contains a listing of TV channels and station icons that will be presented on the Listview button. FIG. 7 presents a sample of the first few rows of this file. Refer to page 19 to view the entire file.

	A	B	C	D
1	NAME	CHANNEL CODE	ICON	RATING
2	A&E	118	A&E.jpg	PG-13
3	ABC	8	ABC.jpg	PG-13
4	ABCFAM	180	ABCFAM.jpg	PG-13
5	ANIMAL	184	ANIMAL.jpg	PG-13
6	BBC	135	BBC.jpg	PG-13
7	BRAVO	129	BRAVO.jpg	PG-13
8	CBS	11	CBS.jpg	PG-13
9	CMDY-E	107	CMDY-E.jpg	PG-13
10	CMT	166	CMT.jpg	PG-13
11	CNBC	208	CNBC.jpg	PG-13
12	CNN	200	CNN.jpg	PG-13
13	CW	33	CW.jpg	PG-13

channelList.csv has four columns with headers: NAME, CHANNEL CODE, ICON and RATING

FIG. 7 Data Source File - "channelList.csv" (CSV file with headers)

1. In NetLinx Studio 4, establish communication with the Master (refer to NetLinx Studio 4 online help for details).
2. Select **Tools > File Transfer** to open the *File Transfer* dialog (FIG. 8):

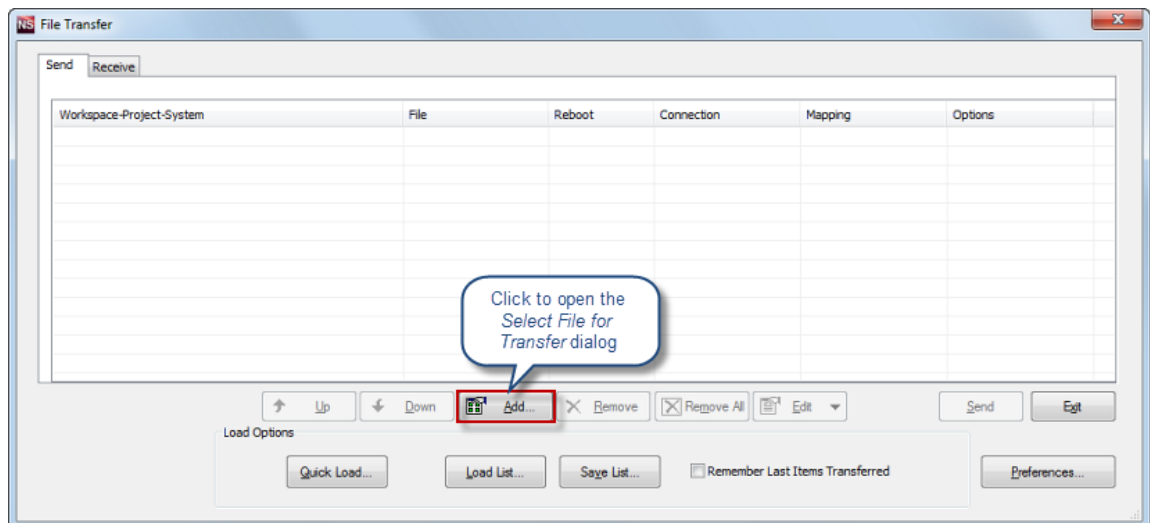


FIG. 8 NetLinx Studio 4 - File Transfer dialog

3. Click **Add** to open the *Select Files for File Transfer* dialog, and open the *Other* tab (FIG. 9):

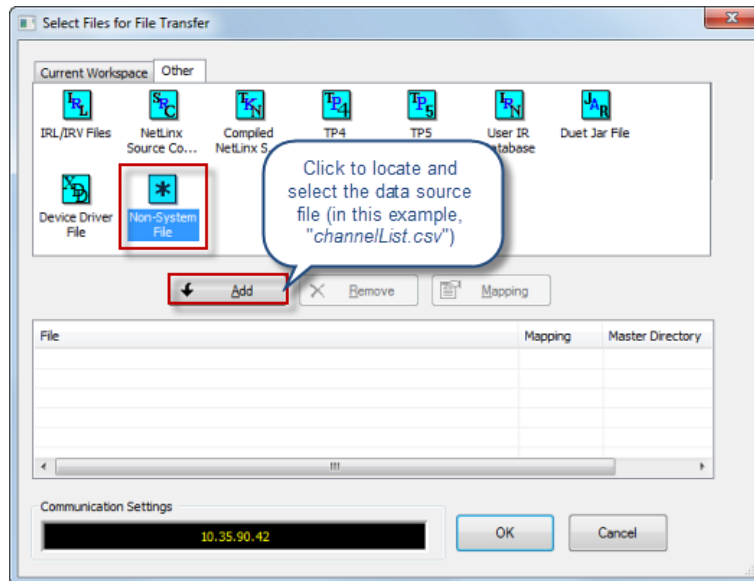


FIG. 9 NetLinx Studio 4 - Select Files for File Transfer dialog

4. Select **Non-System File**, then click **Add**.
5. In the *Open* dialog, locate and select the "channelList.csv" file and click **Open** to access the *Enter Device Mapping Information* dialog (FIG. 10).

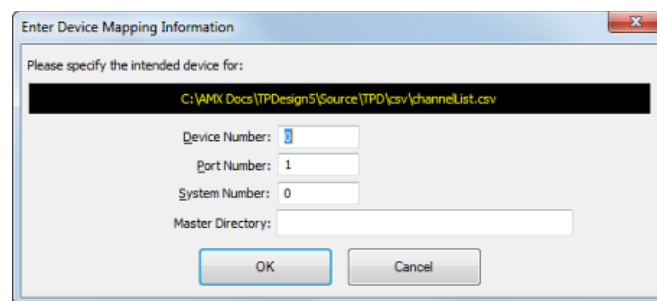


FIG. 10 NetLinx Studio 4 - Enter Device Mapping Information dialog

- a. Enter the *Device*, *Port* and *System* Number for the target NX Master.
- b. In the *Master Directory* field, enter the name of the directory on the NX Master that contains the data source file.



*If no directory is specified in the Master Directory field, the file will be copied to the root directory on the Master.*

- c. Click **OK** to save changes and return to the *Select Files For File Transfer* dialog.
6. In the *Select Files For File Transfer* dialog, the selected file and its device information are indicated in the *Files* list (FIG. 11):

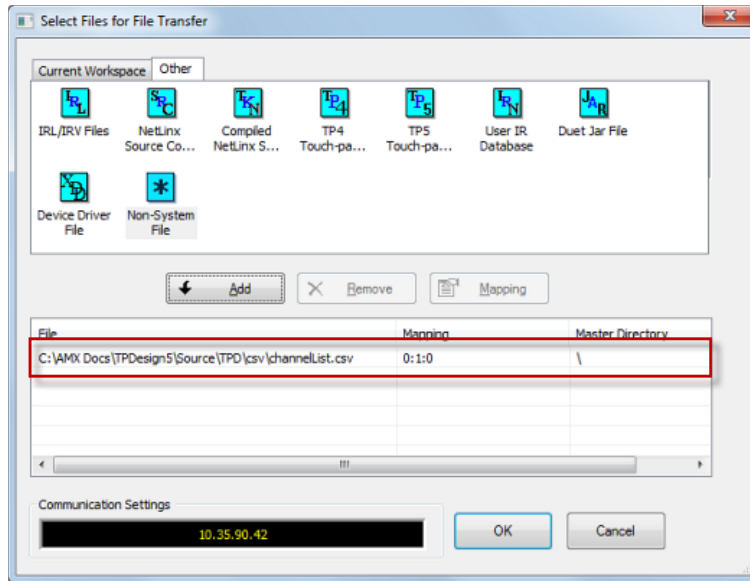


FIG. 11 NetLinx Studio 4 - Select Files for File Transfer dialog indicating a CSV file for transfer

7. Click **OK** to close this dialog and return to the *File Transfer* dialog.
8. Click **Send** to initiate the file transfer. The program will indicate when the transfer is complete.

#### 4) Add the Dynamic Data Source to the Project

To add the data source file (channelList.csv) to the TPDesign5 project:

1. Open the Resource Manager to the *Dynamic Data Sources* tab and click **New** to open the *Create Dynamic Data Source* dialog (FIG. 12):

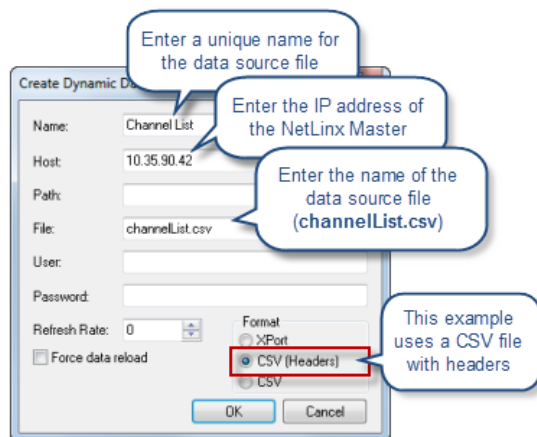
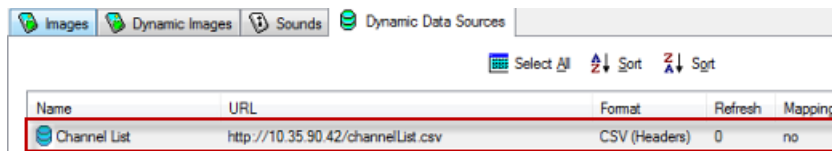


FIG. 12 Create Dynamic Data Source dialog with Example data (ChanelList.csv)

2. In the *Name* field, enter a unique friendly name for the data source. For this example, enter "**Channel List**".
3. In the *Host* field, enter the host name, which must be a fully qualified DNS or IP address.
4. In the *File* field, enter a file name that indicates the full path to the location of the source file.
5. In the *User* field, enter the user name required by the NX Master or server for authentication (if required).
6. In the *Password* field, enter the password required by the NX Master or server for authentication (if required).
7. In the *Refresh Rate* field, use the up/down arrows to adjust the number of seconds between refreshes in which the resource is downloaded again. Refreshing resources will cause the button displaying that resource to refresh as well. The default value is 0, which means that the resource is only downloaded once.
8. Under *Format*, select **CSV (Headers)**, since the data source file in this example (channelList.csv) uses a CSV file with headers.

9. Click **OK** to save changes and close this dialog. The new data source is indicated in the Resource Manager - Dynamic Data Sources tab (FIG. 13):



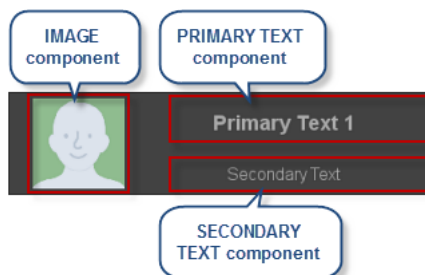
**FIG. 13** Resource Manager - Dynamic Data Sources tab indicating "channelList.csv" as the data source



The Listview button in the TVGuide.TP5 file is pre-configured to use ChannelList (channelList.csv) as its data source file. However, it is necessary to update the Host address with the IP address of your NX Master as shown above. Double-click on Channel List in the Resource Manager to open the Edit Dynamic Data Source dialog and update accordingly.

### 5) Map the Data from the Data Source File to the Listview Button Components

It is necessary to map the data in the *channelList.csv* file to the three fields that comprise the Listview button layout. These three fields (called Components in TPDesign5) are: *Primary Text*, *Secondary Text* and *Image* (FIG. 14):



**FIG. 14** Listview Button Components

#### Step One: Analyze the Data Source

It is necessary to understand the contents of the data source file in order to map the data to the Components in the Listview button. In this example, the *channelList.csv* file contains four columns with headers: *NAME*, *CHANNEL*, *ICON* and *RATING* (FIG. 15):

	A	B	C	D
	NAME	CHANNEL CODE	ICON	RATING
1	A&E	118	A&E.jpg	PG-13
2	ABC	8	ABC.jpg	PG-13
3	ABCFAM	180	ABCFAM.jpg	PG-13
4	ANIMAL	184	ANIMAL.jpg	PG-13
5	BBC	135	BBC.jpg	PG-13
6	BRAVO	129	BRAVO.jpg	PG-13
7	CBS	11	CBS.jpg	PG-13
8	CMDY-E	107	CMDY-E.jpg	PG-13
9	CMT	166	CMT.jpg	PG-13
10	CNBC	208	CNBC.jpg	PG-13
11	CNN	200	CNN.jpg	PG-13
12	CW	33	CW.jpg	PG-13

**FIG. 15** Understanding the contents of the data source file - channelList.csv

In this example:

- The items in the **NAME** column will be mapped to display as the *Primary Text* component of the list items in the Listview button.
- The items in the **RATING** column will be mapped to display as the *Secondary Text* component of the list items in the Listview button.



- The items in the **ICON** column will be mapped to display as the *Image* component of the list items in the Listview button.
- The items in the **CHANNEL CODE** column will not be mapped to display in the Listview button. However, this data can still be put to use in a custom event - see 8) *Write a Custom Event To Respond To User Selection* on page 15 for details.

### Step Two: Map the Data to Components of the Listview button

1. With the Listview button selected, open the Resource Manager to the *Dynamic Data Sources* tab.
2. Select the data source (*channelList.csv*) that is assigned to the Listview button (as described on page 9):

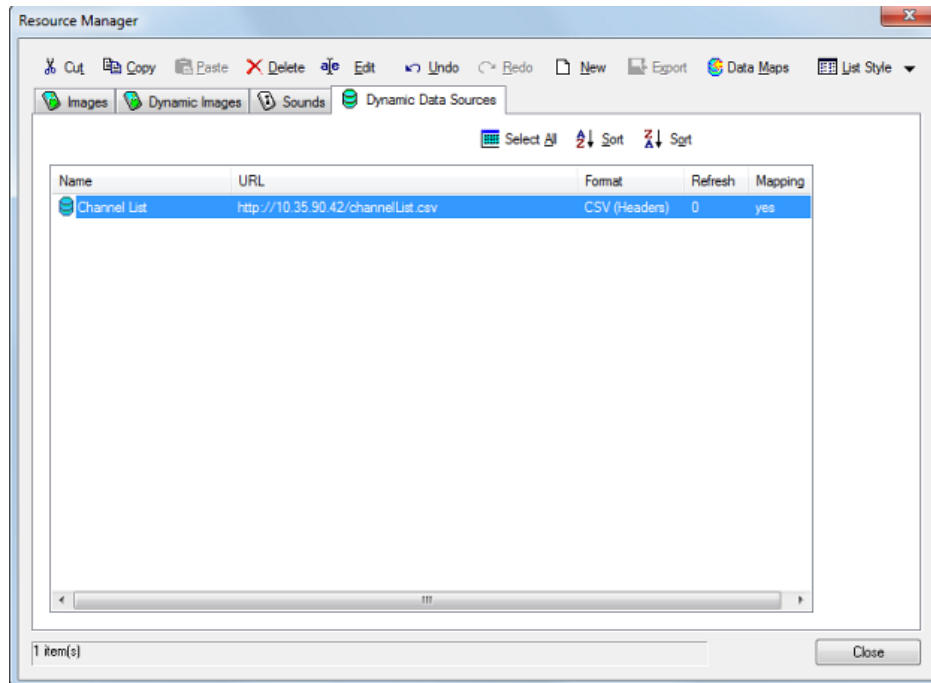


FIG. 16 Resource Manager - Dynamic Data Source tab

3. Click the **Data Maps** button to access the *Dynamic Data Mappings - Listview Buttons* dialog (FIG. 17):

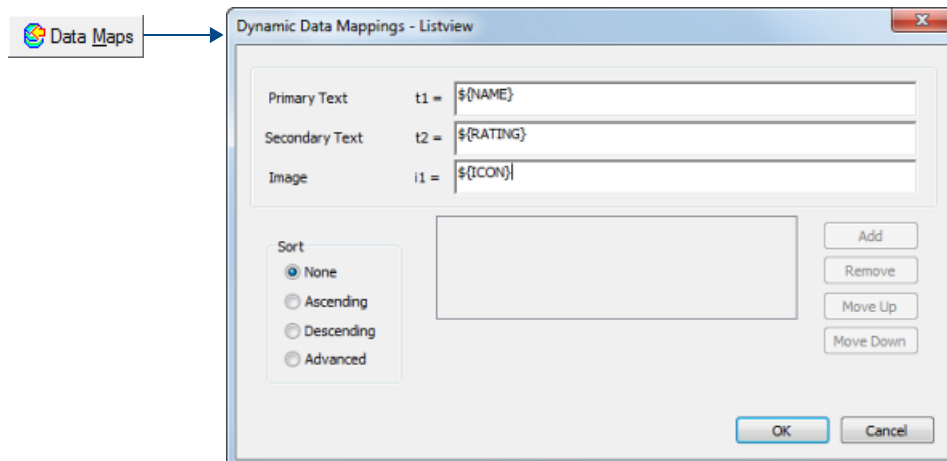


FIG. 17 Dynamic Data Mappings - Listview dialog (with example data indicated)

4. Use the fields in this dialog to specify the device mapping for the selected Listview button and the selected Data Source (see *Dynamic Data Mappings - Syntax Requirements (CSV with Headers)* below).



The Listview button in the TV Guide demo is pre-configured with the data mapping settings shown above.

### Dynamic Data Mappings - Syntax Requirements (CSV with Headers)

Note that the syntax requirements for these fields depends on the type of file used as the data source. The data source file in this example uses a CSV file with headers. The syntax requirements for data mapping to a CSV with headers is described below:

Dynamic Data Mappings - Syntax Requirements (CSV with Headers)	
<b>Primary Text:</b>	
For CSV files with headers, the syntax is:	
<b><code>\${header}</code></b>	
Following this syntax, enter the name of the header in the data source file to be displayed as the Primary Text component of the Listview button. In this example, channelList.csv lists TV channel names in the "NAME" column. To display the contents of the NAME column as the Primary Text component, enter <b><code>\${NAME}</code></b> in the <i>Primary Text</i> field:	
Primary Text	t1 = <code>\${NAME}</code>
<b>Secondary Text:</b>	
For CSV files with headers, the syntax is:	
<b><code>\${header}</code></b>	
Following this syntax, enter the name of the header in the data source file to be displayed as the Secondary Text component of the Listview button. In this example, channelList.csv lists TV channel ratings in the "RATING" column. To display the contents of the RATING column as the Secondary Text component, enter <b><code>\${RATING}</code></b> in the <i>Secondary Text</i> field:	
Secondary Text	t2 = <code>\${RATING}</code>
<b>Image:</b>	
For CSV files with headers, the syntax is	
<b><code>\${header}</code></b>	
Following this syntax, enter the name of the header in the data source file to be displayed as the Image component of the Listview button. In this example, channelList.csv lists the file names of the image files associated with each TV channel (station icons) in the "ICON" column. To display the contents of the ICON column as the Image component, enter <b><code>\${ICON}</code></b> in the <i>Image</i> field:	
Image	i1 = <code>\${ICON}</code>



The fields in the Dynamic Data Mappings - Listview Buttons dialog are case-sensitive.

### 6) Add Image Files to the Project

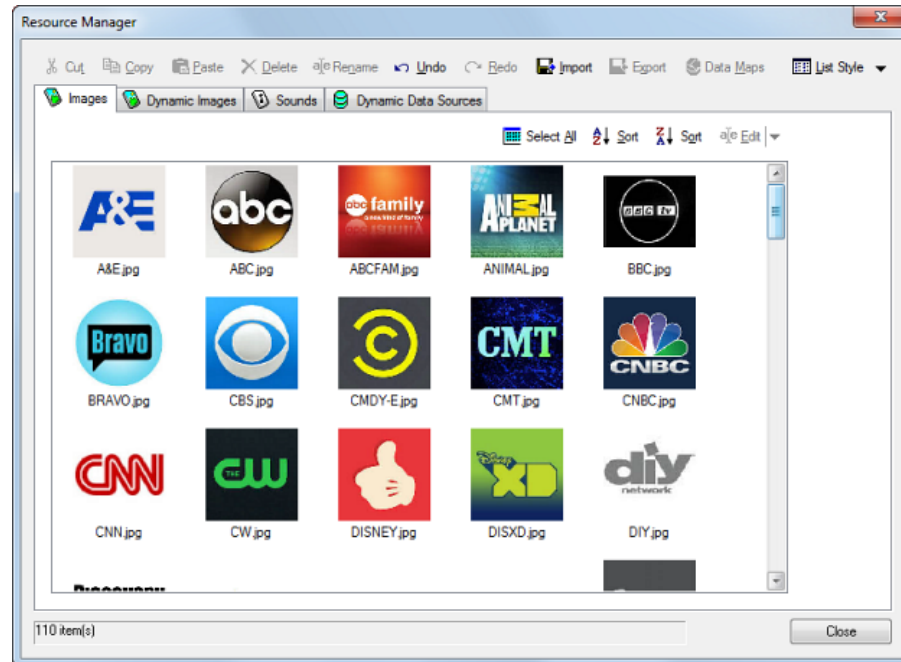
In the data source file for this example (channelList.csv), the *ICON* column lists image files associated with each TV channel in the list (FIG. 18):

	A	B	C	D
1	NAME	CHANNEL CODE	ICON	RATING
2	A&E	118	A&E.jpg	PG-13
3	ABC	8	ABC.jpg	PG-13
4	ABCFAM	180	ABCFAM.jpg	PG-13
5	ANIMAL	184	ANIMAL.jpg	PG-13
6	BBC	135	BBC.jpg	PG-13
7	BRAVO	129	BRAVO.jpg	PG-13
8	CBS	11	CBS.jpg	PG-13
9	CMDY-E	107	CMDY-E.jpg	PG-13
10	CMT	166	CMT.jpg	PG-13

FIG. 18 Data Source File - "channelList.csv"

In order to display these image files on the Listview button, the image files named in the data source file must be added to the project, via the Resource Manager - Images tab:

1. Open the Resource Manager to the *Images* tab.
2. Click **Import** to access the *Open* dialog. Locate and select all of the image files that are named in the data source file (channelList.csv).
3. Click **OK** to import the selected files and return to the Resource Manager (FIG. 19).



**FIG. 19** Resource Manager Images tab - Channel images imported

4. Click **Close** to close the Resource Manager.



*The TVGuide.TP5 file in the TV Guide demo has the channel images shown above already imported into the project. These image files are also available in the "Channel images" folder (included in the TV Guide.ZIP file).*

## 7) Assign a Data Source file to the Listview Button

The data source (channelList.csv) is associated with the Listview button via the *Dynamic Data Source* property (in the *General* tab of the Properties window):

1. With the Listview button selected, click the browse button in the **Dynamic Data Source** (General) property to open the *Select Resource* dialog (FIG. 20):

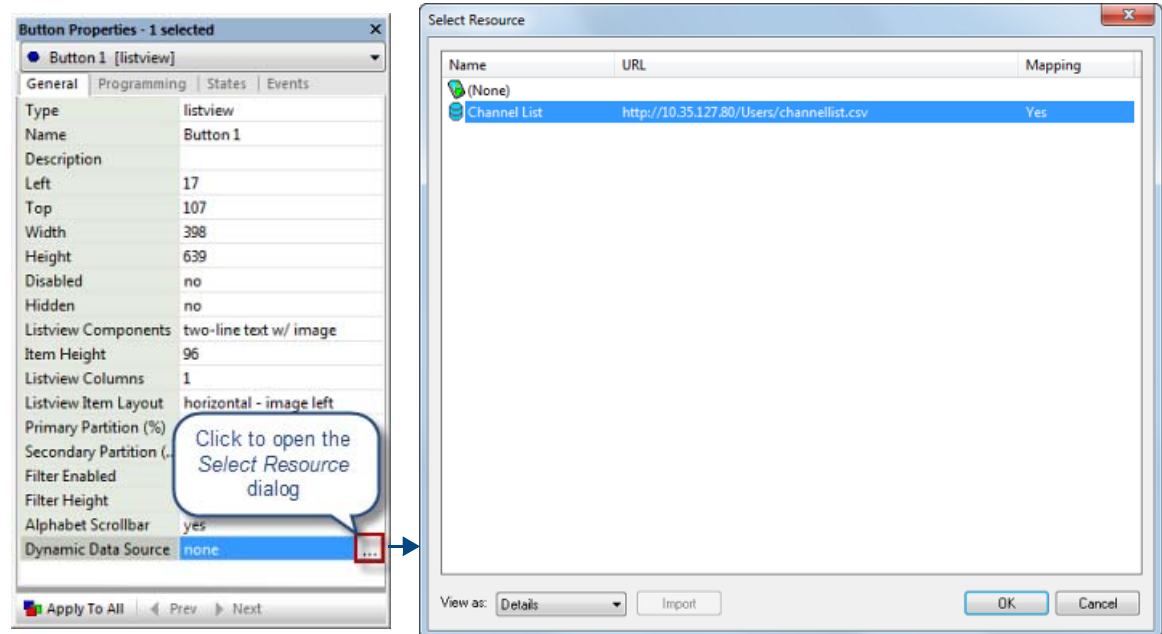


FIG. 20 Dynamic Data Source (General) Property and Select Resource dialog

2. Select the CSV file to use as the data source (in this example, "channelList.csv").
3. Click **OK** to close this dialog.
4. The selected Data Source file is indicated in the *Dynamic Data Source* property (see "Channel List" in FIG. 21):

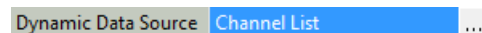


FIG. 21 Dynamic Data Source property indicating "channelList.csv"



NOTE

The "TVGuide.TP5" file included in the TV Guide demo has "Channel List" already assigned as the Dynamic Data Source for the Listview button.

## 8) Write a Custom Event To Respond To User Selection

When the user selects an item on the Listview button, the entire record for that selection is sent to the NX Master. A `CUSTOM_EVENT` is raised and within this function the desired information can be retrieved for the selection. In this example, the channel number needs to be retrieved. The channel number can then be used to initiate a channel change on the cable box. Note that in the *channelList.csv* file, the channel numbers are listed in the *CHANNEL CODE* column (FIG. 22):

	A	B	C	D
1	NAME	CHANNEL CODE	ICON	RATING
2	A&E	118	A&E.jpg	PG-13
3	ABC	8	ABC.jpg	PG-13
4	ABCFAM	180	ABCFAM.jpg	PG-13
5	ANIMAL	184	ANIMAL.jpg	PG-13
6	BBC	135	BBC.jpg	PG-13
7	BRAVO	129	BRAVO.jpg	PG-13
8	CBS	11	CBS.jpg	PG-13
9	CMDY-E	107	CMDY-E.jpg	PG-13
10	CMT	166	CMT.jpg	PG-13
11	CNBC	208	CNBC.jpg	PG-13
12	CNN	200	CNN.jpg	PG-13
13	CW	33	CW.jpg	PG-13

FIG. 22 channelList.csv - CHANNEL CODE column

Listview buttons use the custom event parameter "LISTVIEW\_ON\_ROW\_SELECT\_EVENT" to provide the ability to configure a response to the selection of a list item in a Listview button in NetLinX code. This custom event must be added to the NetLinX code on the NX Master.

1. Use NetLinX Studio 4 to add the following code to the CUSTOM EVENT section of the NetLinX program loaded on the Master:

```
PROGRAM_NAME='TVGuide_CUSTOM_EVENT'
(*****
(*****
(* FILE_LAST_MODIFIED_ON: 04/05/2006 AT: 09:00:25 *)
(*****
(* System Type : NetLinX *)
(*****
(* REV HISTORY: *)
(*****
*)
$History: $
*)
DEFINE_DEVICE
dvTP = 10001:1:0

DEFINE_CONSTANT
// TV Channels Listview button address
INTEGER btnTvListview = 12

DEFINE_VARIABLE

DEFINE_EVENT

// The custom event that is raised whenever a TV listview item is selected on the panel.
// Custom event data has three integers, a data string and an
// encoding string. This is not enough to represent what could
// potentially be a very complex DATA_RECORD. So the listview
// custom event will include a payload ID that can then be used
// to retrieve the contents of the DATA_RECORD.
CUSTOM_EVENT[dvTP,btnTvListview,LISTVIEW_ON_ROW_SELECT_EVENT]
{
    // Variables to hold the ID and type for the payload
    SLONG payloadId
    SLONG payloadType
    // The function to retrieve the payload data takes an array of 1 or more
    // strings that specify which DATA_FIELDS we wish to retrieve. In our
    // example we're interested in 3 fields and 16 characters is long enough
    // to hold the IDs.
    CHAR fields[3][16]
    // Create a DATA_RECORD to hold the retrieved data
    DATA_RECORD record
```

```

// Get the payload ID from the custom event
payloadId = custom.value1
// Get the data type from the custom event
payloadType = custom.value2
// Always check for a valid payload ID and check the payload
// type. Future improvements to the feature may have other
// payload types.
if (payloadId > 0 && payloadType == DATA_STRUCTURE_DATAARECORD)
{
    // Specify which DATA_FIELD IDs we want to retrieve from the payload
    fields[1] = 'NAME'
    fields[2] = 'CHANNEL CODE'
    fields[3] = 'ICON'
    // When retrieving the data, always check the return value. If the
    // return value is greater than zero then the DATA_RECORD that was
    // passed in will be populated with the requested DATA_FIELDS.
    if (DATA_GET_EVENT_RECORD(dvTP, payloadId, fields, record) > 0)
    {
        // Put the channel number and name at the bottom of the TV
        // subpage and show the subpage
        SEND_COMMAND dvTP, "^TXT-13,0,Channel ", record.content[2].value, ' - ', record.content[1].value"
        SEND_COMMAND dvTP, "^BMX-77,0,', record.content[3].value, ',1,10'"
    }
}
}

DEFINE_PROGRAM

( ***** )
( *                END OF PROGRAM                * )
( *                DO NOT PUT ANY CODE BELOW THIS COMMENT                * )
( ***** )

```

2. Compile the code (select **Build > Compile**).
3. Use NetLinX Studio 4 to transfer the AXS file to the NX Master:
  - a. Select **Tools > File Transfer** to open the *File Transfer* dialog.
  - b. In the *Send* tab, click the **Add** button. This opens the *Select Files for File Transfer* dialog.
  - c. In the *Other* tab, select **Non-System File** and click **Add**.
  - d. Select the compiled NetLinX code (in this example, "ISE\_CUSTOM\_EVENT.axs") and click **Open**. This opens the *Enter Device Mapping* dialog.
  - e. Review and edit the D:P:S settings for the target NX Master (leave the *Master Directory* field empty), and click **OK** to close the *Enter Device Mapping* dialog and return to the *Select Files for File Transfer* dialog.
  - f. Select **OK** to return to the *File Transfer* dialog.
  - g. In the *File Transfer* dialog, click **Send** to initiate the file transfer.
  - h. The progress of the transfer is indicated in the Output Bar.



NOTE

The custom event code shown above is included in the NetLinX Studio Workspace file (TV Guide.apw) that is in the TV Guide.ZIP file.

## 9) Transfer the TPDesign5 Project to the Touch Panel

At this point, everything is ready to go: the NX Master has the code to handle custom events and the TPD5 project file is handling the data source/mapping for the Listview button. The only thing left to do is to transfer the TPD5 project containing the Listview button, data source reference and image references to the G5 touch panel:

1. In TPDesign5, select **Transfer > Connect** to open the *Connect* dialog (FIG. 23):

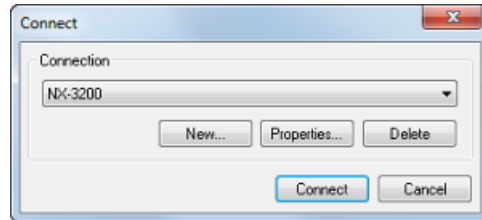


FIG. 23 Connect dialog



*If the Master has never been connected to before, a new connection will need to be configured. Refer to the File Transfer Operations section on page 261 for details.*

2. Select the connection configuration for the target NX Master from the *Connection* drop-down list, and click **Connect**. Once a connection has been established with the Master, select **Transfer > Send to Panel** to open the *Send to Panel* dialog (FIG. 24):

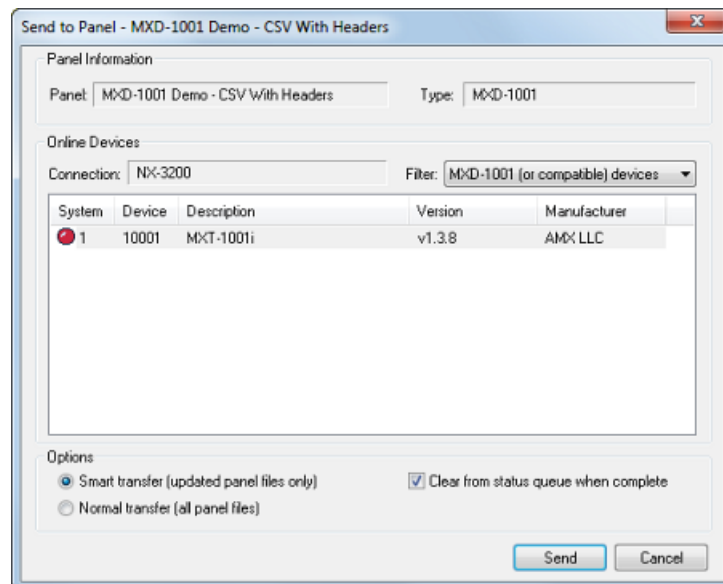


FIG. 24 Send To Panel dialog

3. Click **Send** to begin the file transfer.

When the transfer is complete, the Listview button should appear on the Page it was added to.

### Example 1 (CSV File - With Headers) - Results

FIG. 25 shows an example of a Listview button created by following these steps:

**channelList.csv**

	A	B	C	D
1	NAME	CHANNEL	ICON	RATING
2	A&E	118	A&E.jpg	PG-13
3	ABC	8	ABC.jpg	PG-13
4	ABCFAM	180	ABCFAM.jpg	PG-13
5	ANIMAL	184	ANIMAL.jpg	PG-13
6	BBC	135	BBC.jpg	PG-13
7	BRAVO	120	BRAVO.jpg	PG-13

**Resource Manager (Images tab) showing channel icon images imported into the TPD5 project**

**Resource Manager (Dynamic Data Sources tab) showing the Channel List data source imported into the TPD5 project**

**Data from channelList.csv mapped to Listview components**

**Resulting Listview button as it is displayed on the panel**

**FIG. 25** Example Listview button based on "channelList.csv"

Using the *channelList.csv* file as it's data source:

- It displays each channel's name (based on the data in the NAME column) as the *Primary Text* component.
- It displays each channel's rating (based on the data in the RATING column) as the *Secondary Text* component.
- It displays each channel's station icon (based on the data in the ICON column) as the *Image* component.



While the Listview button shown in this example uses only basic design characteristics, note that Listview buttons support most of the same display options as other button types, including Radiant/Gradient fills, Text Effects, Opacity, etc... Use these options to create eye-catching designs, just like for any other button type.



**Reference: "channelList.csv" (CSV File With Headers)**

NAME, CHANNEL CODE, ICON, RATING

A&E, 118, A&E.jpg, PG-13  
 ABC, 8, ABC.jpg, PG-13  
 ABCFAM, 180, ABCFAM.jpg, PG-13  
 ANIMAL, 184, ANIMAL.jpg, PG-13  
 BBC, 135, BBC.jpg, PG-13  
 BRAVO, 129, BRAVO.jpg, PG-13  
 CBS, 11, CBS.jpg, PG-13  
 CMDY-E, 107, CMDY-E.jpg, PG-13  
 CMT, 166, CMT.jpg, PG-13  
 CNBC, 208, CNBC.jpg, PG-13  
 CNN, 200, CNN.jpg, PG-13  
 CW, 33, CW.jpg, PG-13  
 DISNEY, 172, DISNEY.jpg, G  
 DISXD, 174, DISXD.jpg, PG  
 DIY, 111, DIY.jpg, G  
 DSC, 182, DSC.jpg, PG  
 ENC, 340, ENC.jpg, PG-13  
 ESPN, 140, ESPN.jpg, PG-13  
 ESPN2, 144, ESPN2.jpg, PG-13  
 ESQTV, 191, ESQTV.jpg, PG-13  
 FOOD, 110, FOOD.jpg, PG  
 FOX, 205, FOX.jpg, PG-13  
 FUSE, 164, FUSE.jpg, PG-13  
 FXM, 384, FXM.jpg, R  
 FXX, 390, FXX.jpg, R  
 FYI, 119, FYI.jpg, PG-13  
 GOLF, 401, GOLF.jpg, G  
 GSN, 116, GSN.jpg, PG  
 HALMRK, 185, HALMRK.jpg, R  
 HBO2e, 301, HBO2e.jpg, R  
 HBOe, 300, HBOe.jpg, PG  
 HGTV, 112, HGTV.jpg, PG-13  
 ID, 192, ID.jpg, PG-13  
 IFC, 133, IFC.jpg, PG-13  
 ION, 216, ION.jpg, PG-13  
 LIF-E, 108, LIF-E.jpg, PG-13  
 LMN, 109, LMN.jpg, PG-13  
 MAXe, 315, MAXe.jpg, R  
 MNT, 27, MNT.jpg, PG-13  
 msnbc, 209, MSNBC.jpg, PG-13  
 MTV-E, 160, MTV-E.jpg, R  
 NBC, 5, NBC.jpg, PG-13  
 NGC, 186, NGC.jpg, PG-13  
 NIK, 170, NIK.jpg, PG-13  
 OWN, 186, OWN.jpg, R  
 OXYGN, 127, OXYGN.jpg, R  
 PBS, 13, PBS.jpg, PG-13  
 QVC, 137, QVC.jpg, G  
 REELZ, 299, REELZ.jpg, PG-13  
 SCI, 193, SCI.jpg, R  
 SHO, 318, SHO.jpg, R  
 SPIKE, 241, SPIKE.jpg, R  
 STARZ, 350, STARZ.jpg, PG-13  
 SUNDe, 358, SUNDe.jpg, PG-13  
 SYFY, 122, SYFY.jpg, PG-13  
 TBS, 139, TBS.jpg, PG-13  
 TCM, 132, TCM.jpg, PG-13  
 THC, 120, THC.jpg, PG-13  
 TLC, 183, TLC.jpg, PG-13  
 TMC, 132, TMC.jpg, PG-13  
 TNT, 138, TNT.jpg, PG-13  
 TOON, 176, TOON.jpg, PG-13  
 TRAVEL, 196, TRAVEL.jpg, PG-13  
 truTV, 149, TRU.jpg, PG-13  
 TVLAND, 106, TVLAND.jpg, PG-13  
 USA-E, 105, USA.jpg, R  
 VH1, 162, VH1.jpg, R  
 WE, 128, WE.jpg, PG-13  
 WGN, 239, WGN.jpg, PG-13

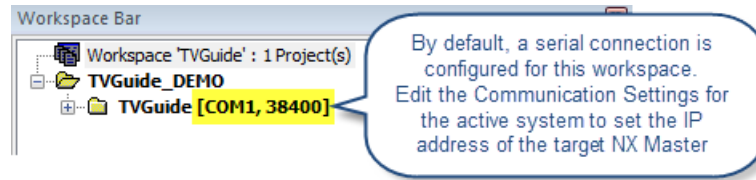
### TV Guide Demo File ("TVGuide.ZIP")

Demo (ZIP) files for the Listview examples presented here are available to download from the UI RESOURCE CENTER at [www.amx.com](http://www.amx.com). The preceding example followed the *TV Guide* demo. The TV Guide demo ZIP file (**TVGuide.ZIP**) contains the following:

TVGuide.ZIP Contents	
File	Description
<b>channelList.csv</b>	This CSV file (with headers) will be used as the data source file for this example.
<b>TVGuide.TP5</b>	TPDesign5 project file that includes a Listview button pre-configured to use the layout properties and data source file shown in the <i>Listview Button/Dynamic Data Example 1: CSV File - With Headers</i> example (see page 19).
<b>"channel images" folder</b>	This folder contains the images used for the Listview button in this example.
<b>TVGuide.apw</b>	NetLinX Studio 4 Workspace file, with the Listview demo custom event defined. This Workspace contains the following files: <ul style="list-style-type: none"> <li>• TVGuide_CUSTOM_EVENT.axs</li> <li>• TVGuide_CUSTOM_EVENT.src</li> <li>• TVGuide_CUSTOM_EVENT.tkn</li> <li>• TVGuide_CUSTOM_EVENT.tko</li> </ul>

To use this demo:

1. Download the *TVGuide.ZIP* file and extract it's contents to a known location.
2. Launch TPDesign5 and open the *TVGuide.TP5* project file. Use TPDesign5 to set to the Host (IP) address for the data source file:
  - a. Open the Resource Manager to the *Dynamic Data Sources* tab, and double-click on the **channelList.csv** file to access the *Edit Dynamic Data Source* dialog.
  - b. Edit the **Host** field with the IP address of the NX Master that will host the file. Click **OK** to save changes and close this dialog.
  - c. Close the Resource Manager.
  - d. Save changes and close the TP5 project.
3. Use NetLinX Studio 4 to transfer the *channelList.csv* data source file to the target NX Master.
  - a. Launch NetLinX Studio 4 and select **Tools > File Transfer** to open the *File Transfer* dialog (*Send* tab).
  - b. Click **Add** to open the *File Transfer* dialog, and open the **Other** tab.
  - c. Select *Non-System File* and click **Add** to access the *Open* dialog.
  - d. Locate and select the *channelList.csv* file and click **Open**
  - e. In the *Enter Device Mapping Information* dialog, review (and edit if necessary) the mapping information for this file, and click **OK** to return to the *Select File for File Transfer* dialog.
  - f. Click **OK** to return to the *File Transfer* dialog. Verify that the *channelList.csv* file is selected for transfer, and click **Send**. Refer to 3) *Host the Data Source File (CSV with Headers) on the NX Master* on page 7 for more details.
4. In NetLinX Studio 4, open the *TVGuide.apw* workspace file (**File > Open Workspace**).  
This Workspace contains NetLinX source code that is pre-configured with a Custom Event for user selection, as well as a TPDesign5 project that includes a pre-configured Listview button that uses *channelList.csv* as it's data source.
5. Build the Workspace: Select **Build > Build Active System**.
6. Transfer all files contained in the Workspace to the target NX Master:
  - a. Select **Settings > Active System Communication Settings** to open the *Communication Settings* dialog. Use the options in this dialog to establish a connection to the target NX Master. Note that by default, the workspace is configured to use Serial communication (FIG. 26):



**FIG. 26** NetLinx Studio 4 Workspace Bar - TV Guide demo (default communication settings)

This IP address should be the same as was specified for the data source file (see Step 3 above). See NetLinx Studio 4 online help for details on configuring communication settings.

- b.** Select **Tools > File Transfer** to open the *File Transfer* dialog (*Send* tab). Remove any files (from previous transfer operations) that may be in the list.
- c.** Click **Add** to open the *Select Files for File Transfer* dialog (*Current Workspace* tab).
- d.** Click the top-level *Projects* directory to auto-select all files in the Workspace.
- e.** Verify that the IP address indicated here indicates the correct NX Master, and click **OK** to save changes and return to the *File Transfer* dialog.
- f.** In the *File Transfer* dialog, click **Send** to transfer the Workspace files to the target NX Master.



# Example 2: CSV File - No Headers

## Overview

The following instructions describe using the Conference Rooms demo for creating a Listview button with a dynamic data source in the form of a CSV file *without* headers.



*This set of instructions uses files that are included in the "Conference Rooms.ZIP" demo file which is available to download from the UI RESOURCE CENTER at [www.amx.com](http://www.amx.com).*

The resulting Listview button will display a listing of Conference rooms with each room's name, phone number and room icons (FIG. 27):

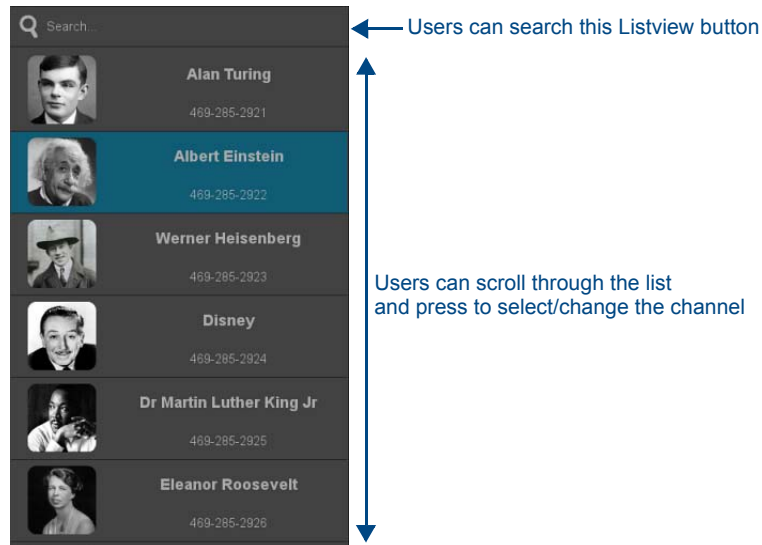


FIG. 27 Example - Listview button based on "conference.csv"

## Before You Begin

1. Download the *Conference.ZIP* file from the UI RESOURCE CENTER at [www.amx.com](http://www.amx.com) and extract it's contents to a known location.
2. Open the *conference.csv* file and analyze it's contents. It is a relatively simple csv file that consists of three columns without headers. See page 37 to view this file.

### 1) Create (draw) a Listview button

1. In TPDesign5, open a Page and use the Button Draw tool to create a new button.
2. With the new button selected, click the **Type** (General) property and select **Listview** from the drop-down of button types. This selection sets the new button as a Listview button, and enables a set of Listview-specific properties (FIG. 28):

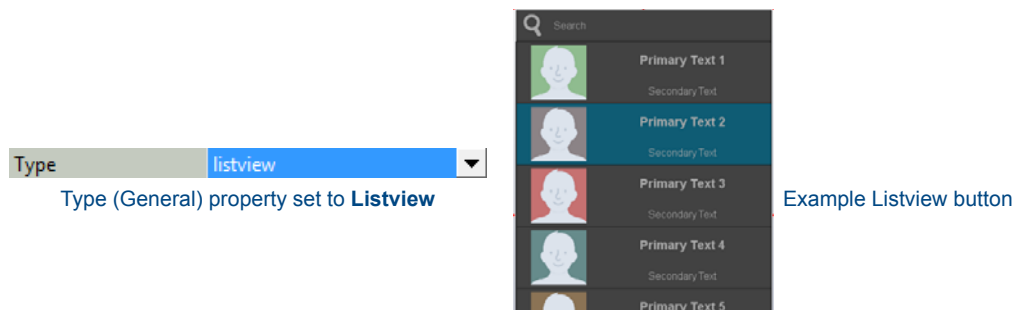


FIG. 28 Type (General) Property set to Listview



The "CSV.TP5" file included in the Conference Rooms demo has a Listview button already drawn on the "Main" page.

2) Review the Listview Button Properties

Use the options in the Properties window to view/edit the *General*, *Programming* and *States* properties for the Listview button to match the settings shown in FIG. 29:

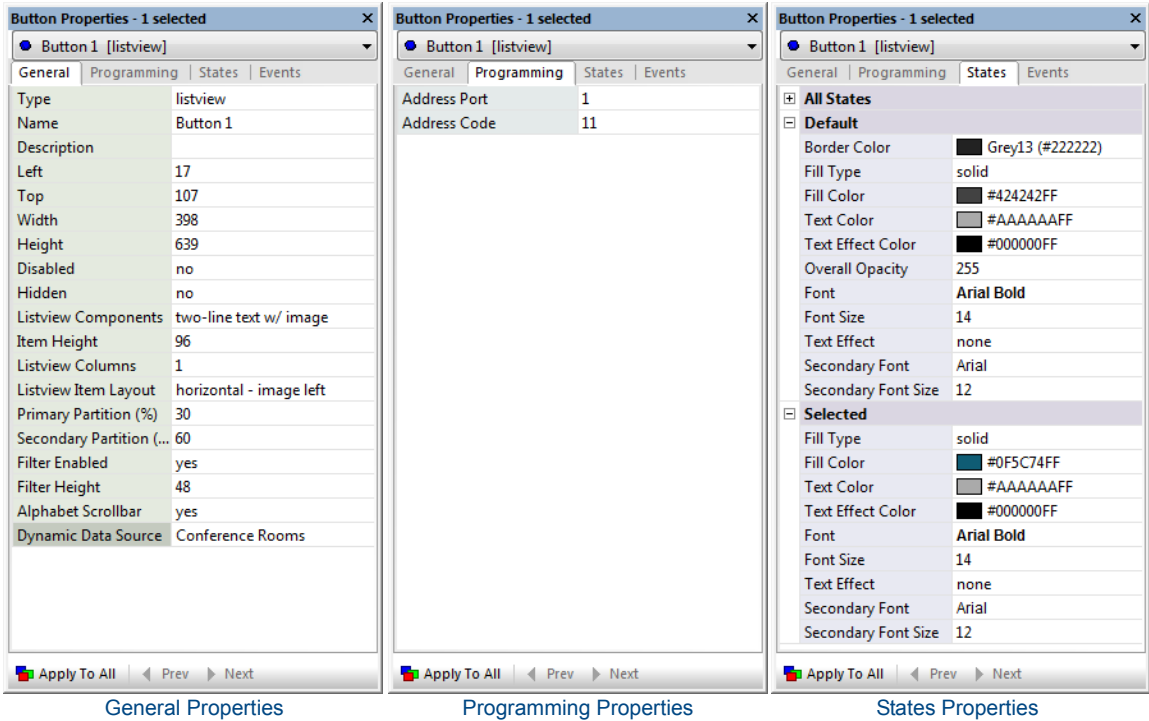


FIG. 29 Properties for the Conference Rooms Listview Button



The Listview button in the Conference Rooms demo is pre-configured with the General, Programming and States properties shown above.

Refer to the *Working With Listview Button Properties* section on page 3 for details on Listview-specific button properties.

### 3) Host a Data Source File (CSV without Headers) on the NX Master

In this example, "conference.csv" will be the data source for the Listview button. This CSV file will be hosted on the NX Master. The "conference.csv" file contains a listing of conference rooms with phone numbers and room icons that will be presented on the Listview button. FIG. 30 presents a sample of the first few rows of this file. Refer to page 37 to view the entire file.

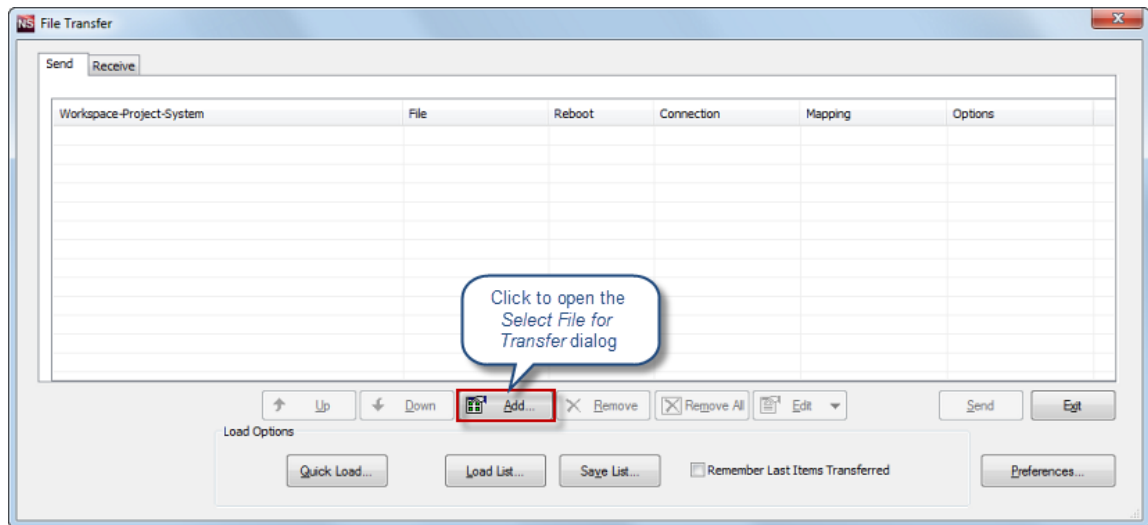
	A	B	C
1	Alan Turing	TURING.jpg	469-285-2921
2	Albert Einstein	EINSTEIN.jpg	469-285-2922
3	Werner Heisenberg	HEISENBERG.jpg	469-285-2923
4	Disney	WALTDISNEY.jpg	469-285-2924
5	Dr Martin Luther King	MARTINLUTHER.jpg	469-285-2925
6	Eleanor Roosevelt	ELEANOR.jpg	469-285-2926
7	Emmeline Pankhurst	PANKHURST.jpg	469-285-2927
8	Frank Sinatra	FRANKSINATRA.jpg	469-285-2928
9	Henry Ford	HENRYFORD.jpg	469-285-2929
10	Jackie Robinson	JACKIEROBINSON.jpg	469-285-2930
11	Jean Piaget	PIAGET.jpg	469-285-2931
12	Margaret Thatcher	MARGARETHATCHER.jpg	469-285-2932

conference.csv has three columns with no headers.  
Column 1 contains the room names  
Column 2 contains the room icons  
Column 3 contains the room phone numbers

**FIG. 30** Data Source File - "conference.csv" (CSV file without headers)

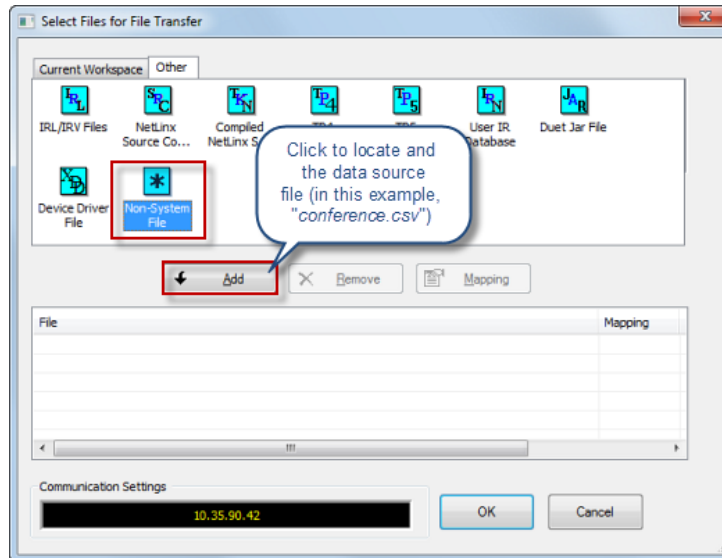
To host a CSV file on the NX Master:

1. In NetLinx Studio 4, establish communication with the Master (refer to NetLinx Studio 4 online help for details).
2. Select **Tools > File Transfer** to open the *File Transfer* dialog (FIG. 31):



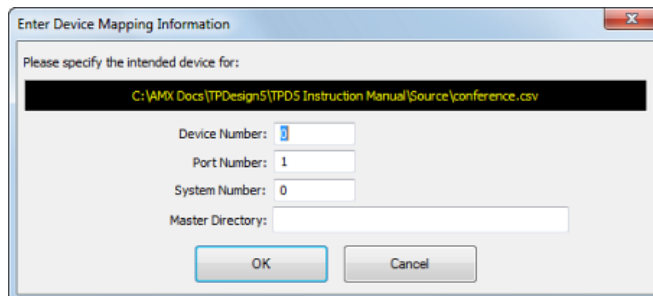
**FIG. 31** NetLinx Studio 4 - File Transfer dialog

3. Click **Add** to open the *Select Files for File Transfer* dialog, and open the *Other* tab (FIG. 32):



**FIG. 32** NetLinx Studio 4 - Select Files for File Transfer dialog

4. Select **Non-System File**, then click **Add**.
5. In the *Open* dialog, locate and select the "conference.csv" file and click **Open** to access the *Enter Device Mapping Information* dialog (FIG. 33).



**FIG. 33** NetLinx Studio 4 - Enter Device Mapping Information dialog

- a. Enter the *Device*, *Port* and *System* Number for the target NX Master.
- b. In the *Master Directory* field, enter the name of the directory on the NX Master that contains the data source file.



*If no directory is specified in the Master Directory field, the file will be copied to the root directory on the Master.*

- c. Click **OK** to save changes and close the *Enter Device Mapping Information* dialog (and return to the *Select Files For File Transfer* dialog).
6. In the *Select Files For File Transfer* dialog, the selected file and its device information are indicated in the *Files* list (FIG. 34):



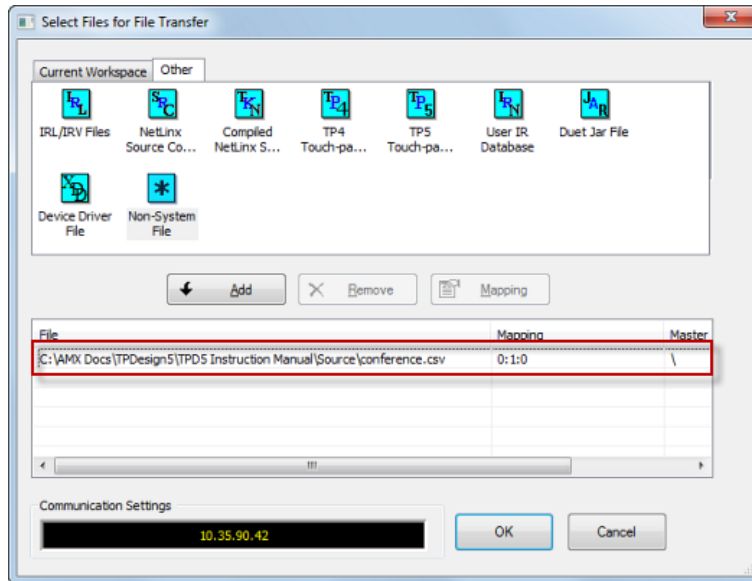


FIG. 34 NetLinx Studio5 - Select Files for File Transfer dialog indicating a CSV file for transfer

7. Click **OK** to close this dialog and return to the *File Transfer* dialog.
8. Click **Send** to initiate the file transfer. The program will indicate when the transfer is complete.

#### 4) Add the Dynamic Data Source to the Project

To add the data source file (chanelList.csv) to the TPDesign5 project:

1. Open the Resource Manager to the *Dynamic Data Sources* tab and click **New** to open the *Create Dynamic Data Source* dialog (FIG. 35):

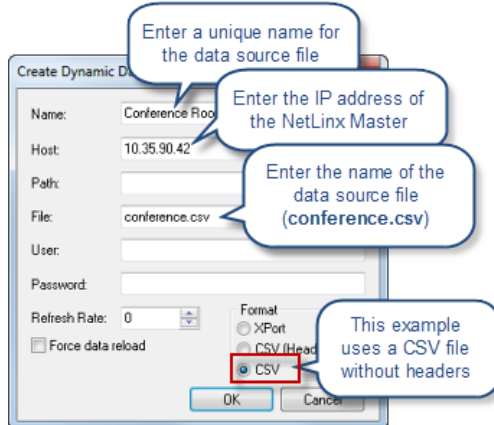
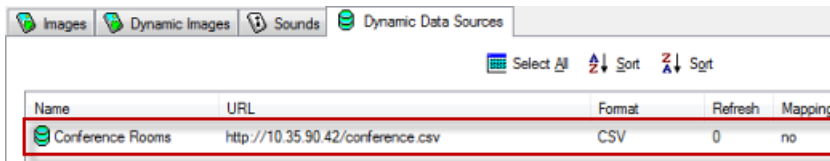


FIG. 35 Create Dynamic Data Source dialog with Example data (conference.csv)

2. In the *Name* field, enter a unique friendly name for the data source. For this example, enter "**Conference Rooms**".
3. In the *Host* field, enter the host name, which must be a fully qualified DNS or IP address.
4. In the *File* field, enter a file name that indicates the full path to the location of the source file.
5. In the *User* field, enter the user name required by the NX Master or server for authentication (if required).
6. In the *Password* field, enter the password required by the NX Master or server for authentication (if required).
7. In the *Refresh Rate* field, use the up/down arrows to adjust the number of seconds between refreshes in which the resource is downloaded again. Refreshing resources will cause the button displaying that resource to refresh as well. The default value is 0, which means that the resource is only downloaded once.
8. Under *Format*, select **CSV**, since the data source file in this example (conference.csv) uses a CSV file without headers.

9. Click **OK** to save changes and close this dialog. The new data source is indicated in the Resource Manager - Dynamic Data Sources tab (FIG. 36):



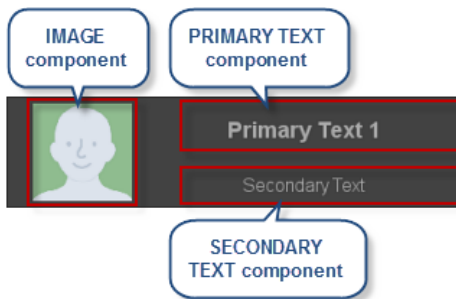
**FIG. 36** Resource Manager - Dynamic Data Sources tab indicating "conference.csv" as the data source



The Listview button in the CSV.TP5 file is pre-configured to use Conference Rooms (conference.csv) as its data source file. However, it is necessary to update the Host address with the IP address of your NX Master as shown above. Double-click on Conference Rooms in the Resource Manager to open the Edit Dynamic Data Source dialog and update accordingly.

### 5) Map the Data from the Data Source File to the Listview Button Components

It is necessary to map the data in the *conference.csv* file to the three fields that comprise the Listview button layout. These three fields (called Components in TPDesign5) are: *Primary Text*, *Secondary Text* and *Image* (FIG. 37):



**FIG. 37** Listview Button - Components

#### Step One: Analyze the Data Source

It is necessary to understand the contents of the data source file in order to map the data to the Components in the Listview button. In this example, the *conference.csv* file contains three columns with no headers.

Note that Column #1 (A) contains room names, Column #2 (B) contains room icons, and Column #3 (C) contains room phone numbers (FIG. 38):

	A	B	C
1	Alan Turing	TURING.jpg	469-285-2921
2	Albert Einstein	EINSTEIN.jpg	469-285-2922
3	Werner Heisenberg	HEISENBERG.jpg	469-285-2923
4	Disney	WALTDISNEY.jpg	469-285-2924
5	Dr Martin Luther King	MARTINLUTHER.jpg	469-285-2925
6	Eleanor Roosevelt	ELEANOR.jpg	469-285-2926
7	Emmeline Pankhurst	PANKHURST.jpg	469-285-2927

**FIG. 38** Understanding the contents of the data source file - channelList.csv

In this example:

- The items in **Column #1** will be mapped to display as the *Primary Text* component of the list items in the Listview button.
- The items in **Column #2** will be mapped to display as the *Image* component of the list items in the Listview button.
- The items in **Column #3** will be mapped to display as the *Secondary Text* component of the list items in the Listview button.

### Step Two: Map the Data to Components of the Listview button

1. With the Listview button selected, open the Resource Manager to the *Dynamic Data Sources* tab.
2. Select the data source (*conference.csv*) that is assigned to the Listview button (as described on page 27):

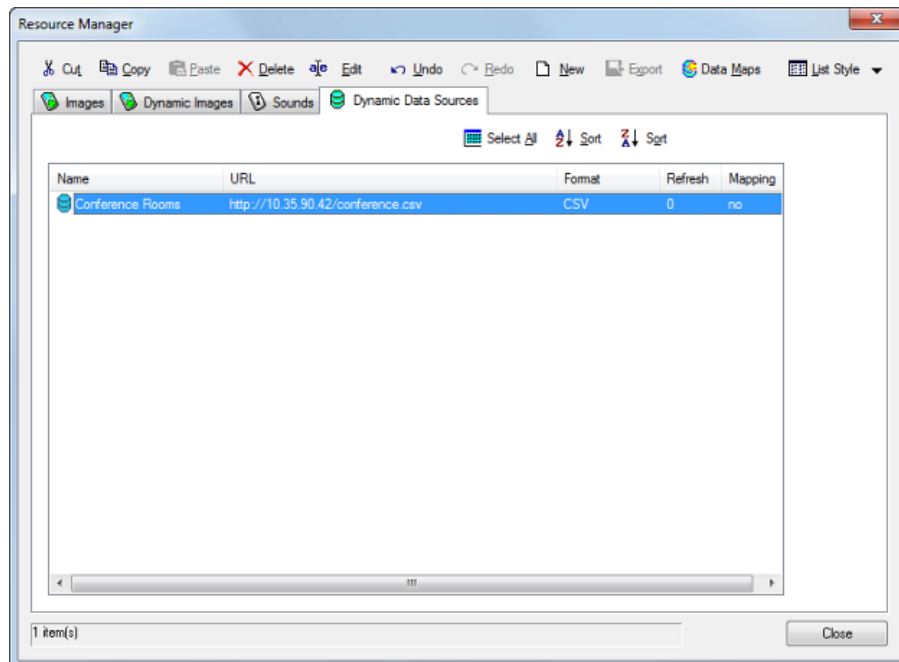


FIG. 39 Resource Manager - Dynamic Data Source tab

3. Click the **Data Maps** button to access the *Dynamic Data Mappings - Listview Buttons* dialog (FIG. 40):

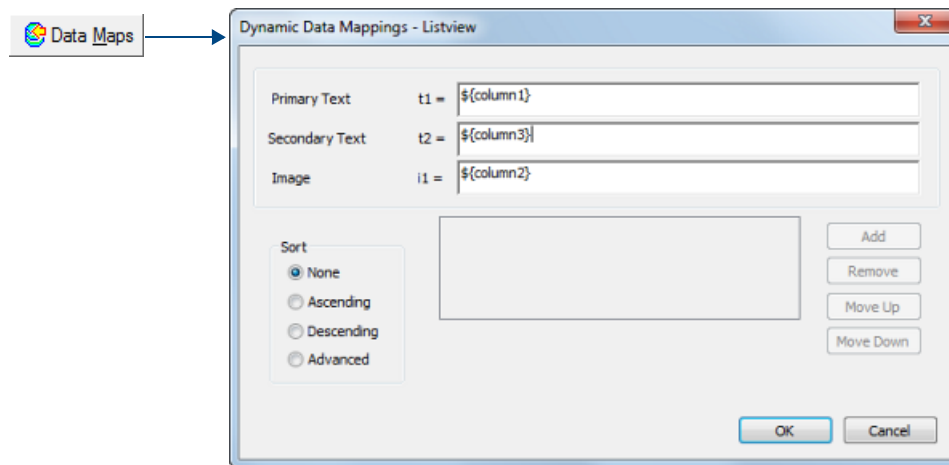


FIG. 40 Dynamic Data Mappings - Listview dialog (with example data indicated)

4. Use the fields in this dialog to specify the device mapping for the selected Listview button and the selected Data Source (see *Dynamic Data Mappings - Syntax Requirements (CSV with Headers)* below).



The Listview button in the Conference Rooms demo is pre-configured with the data mapping settings shown above.

### Dynamic Data Mappings - Syntax Requirements (CSV Without Headers)

Note that the syntax requirements for these fields depends on the type of file used as the data source. The data source file in this example uses a CSV file without headers. In the absence of headers, the columns will be named by default as: column1, column2, column3... The syntax requirements for data mapping to a CSV without headers is described below:

Dynamic Data Mappings - Syntax Requirements (CSV Without Headers)	
<b>Primary Text:</b> For CSV files without headers, the syntax is: <b>`\${column#}`</b> Following this syntax, enter the column # in the data source file to be displayed as the Primary Text component of the Listview button. In this example, conference.csv lists room names in Column #1. To display the contents of Column #1 as the Primary Text component, enter <b>`\${column1}`</b> in the <i>Primary Text</i> field: <div> <div>Primary Text</div> <div>t1 = <input type="text" value="`\${column1}`"/></div> </div>	
<b>Secondary Text:</b> For CSV files without headers, the syntax is: <b>`\${column#}`</b> Following this syntax, enter the column # in the data source file to be displayed as the Secondary Text component of the Listview button. In this example, conference.csv lists room phone numbers in Column #3. To display the contents of Column #3 as the Secondary Text component, enter <b>`\${column3}`</b> in the <i>Secondary Text</i> field: <div> <div>Secondary Text</div> <div>t2 = <input type="text" value="`\${column3}`"/></div> </div>	
<b>Image:</b> For CSV files without headers, the syntax is: <b>`\${column#}`</b> Following this syntax, enter the column # in the data source file to be displayed as the Image component of the Listview button. In this example, conference.csv lists room icons in Column #2. To display the contents of Column #2 as the Image component, enter <b>`\${column2}`</b> in the <i>Image</i> field: <div> <div>Image</div> <div>i1 = <input type="text" value="`\${column2}`"/></div> </div>	



NOTE

*These fields in the Dynamic Data Mappings - Listview Buttons dialog are case-sensitive.*

### 6) Add Image Files to the Project

In the data source file for this example (conference.csv), column #2 lists image files associated with each conference room in the list (FIG. 41):

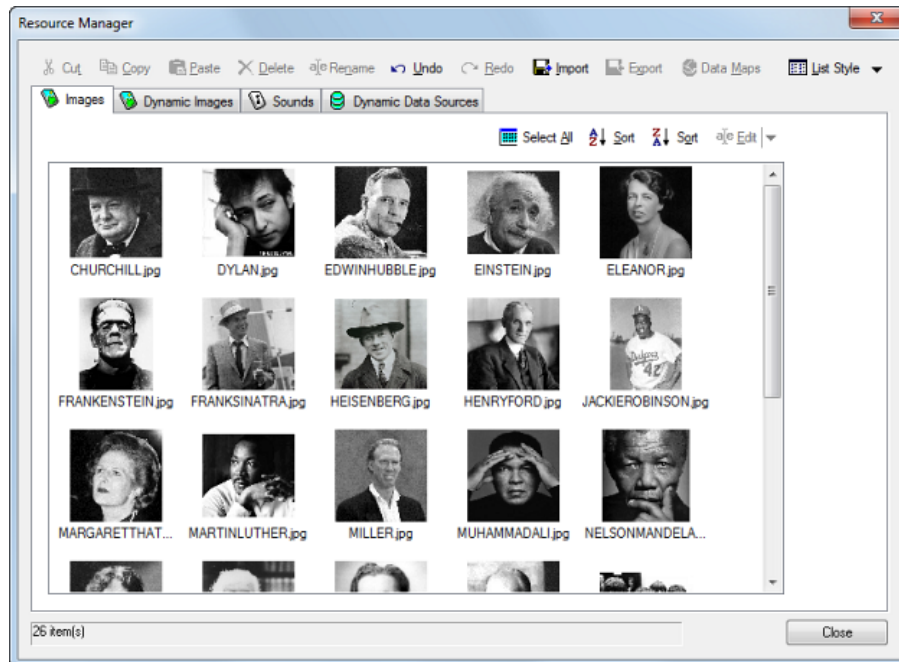
	A	B	C
1	Alan Turing	TURING.jpg	469-285-2921
2	Albert Einstein	EINSTEIN.jpg	469-285-2922
3	Werner Heisenberg	HEISENBERG.jpg	469-285-2923
4	Disney	WALTDISNEY.jpg	469-285-2924
5	Dr Martin Luther King	MARTINLUTHER.jpg	469-285-2925
6	Eleanor Roosevelt	ELEANOR.jpg	469-285-2926
7	Emmeline Pankhurst	PANKHURST.jpg	469-285-2927
8	Frank Sinatra	FRANKSINATRA.jpg	469-285-2928
9	Henry Ford	HENRYFORD.jpg	469-285-2929
10	Jackie Robinson	JACKIEROBINSON.jpg	469-285-2930
11	Jean Piaget	PIAGET.jpg	469-285-2931
12	Margaret Thatcher	MARGARETHATCHER.jpg	469-285-2932
13	Muhammad Ali	MUHAMMADALI.jpg	469-285-2933
14	Nelson Mandela	NELSONMANDELA.jpg	469-285-2934

**FIG. 41** Data Source File - "conference.csv"

In order to display these image files on the Listview button, the image files named in the data source file must be added to the project, via the Resource Manager - Images tab:

1. Open the Resource Manager to the *Images* tab.

2. Click **Import** to access the *Open* dialog. Locate and select all of the image files that are named in the data source file (conference.csv).
3. Click **OK** to import the selected files and return to the Resource Manager (FIG. 42):



**FIG. 42** Resource Manager Images tab - Conference Room images imported

4. Click **Close** to close the Resource Manager.



*The CSV.TP5 file in the Conference Rooms demo has the channel images shown above already imported into the project. These image files are also available in the "conference images" folder (included in the Conference Rooms.ZIP file).*

## 7) Assign a Data Source file to the Listview Button

The data source (conference.csv) is associated with the Listview button via the *Dynamic Data Source* property (in the *General* tab of the Properties window):

1. With the Listview button selected, click the browse button in the **Dynamic Data Source** (General) property to open the *Select Resource* dialog (FIG. 43):

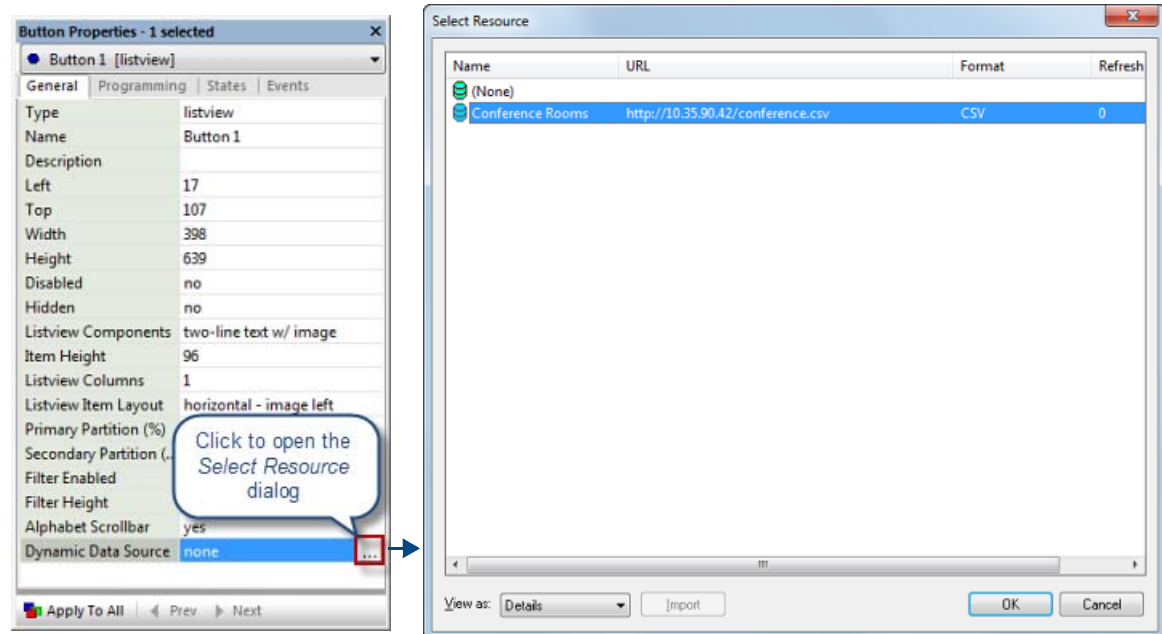


FIG. 43 Dynamic Data Source (General) Property and Select Resource dialog

2. Select the CSV file to use as the data source (in this example, "conference.csv").
3. Click **OK** to close this dialog.
4. The selected Data Source file is indicated in the *Dynamic Data Source* property (see "Conference Rooms" in FIG. 44):

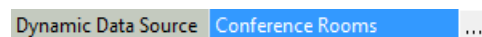


FIG. 44 Dynamic Data Source property indicating "conference.csv"



NOTE

The "CSV.TP5" file included in the Conference Rooms demo has "Conference Rooms" already assigned as the *Dynamic Data Source* for the Listview button.

## 8) Write a Custom Event To Respond To User Selection

When the user selects an item on the Listview button, the entire record for that selection is sent to the NX Master. A `CUSTOM_EVENT` is raised and within this function the desired information can be retrieved for the selection. In this example, the phone number needs to be retrieved. The phone number can then be used to initiate a call to the associated room. Note that in the *conference.csv* file, the phone numbers are listed in Column #3 (FIG. 45):

	A	B	C
1	Alan Turing	TURING.jpg	469-285-2921
2	Albert Einstein	EINSTEIN.jpg	469-285-2922
3	Werner Heisenberg	HEISENBERG.jpg	469-285-2923
4	Disney	WALTDISNEY.jpg	469-285-2924
5	Dr Martin Luther King	MARTINLUTHER.jpg	469-285-2925
6	Eleanor Roosevelt	ELEANOR.jpg	469-285-2926
7	Emmeline Pankhurst	PANKHURST.jpg	469-285-2927
8	Frank Sinatra	FRANKSINATRA.jpg	469-285-2928
9	Henry Ford	HENRYFORD.jpg	469-285-2929
10	Jackie Robinson	JACKIEROBINSON.jpg	469-285-2930
11	Jean Piaget	PIAGET.jpg	469-285-2931
12	Margaret Thatcher	MARGARETHATCHER.jpg	469-285-2932
13	Muhammad Ali	MUHAMMADALI.jpg	469-285-2933
14	Nelson Mandela	NELSONMANDELA.jpg	469-285-2934

**FIG. 45** conference.csv - Column #3 (phone numbers)

Listview buttons use the custom event parameter "LISTVIEW\_ON\_ROW\_SELECT\_EVENT" to provide the ability to configure a response to the selection of a list item in a Listview button in NetLinx code.

This custom event must be added to the NetLinx code on the NX Master.

1. Use NetLinx Studio 4 to add the following code to the CUSTOM EVENT section of the NetLinx program loaded on the Master:

```
PROGRAM_NAME='ISE_CUSTOM_EVENT'
(*****
(*****
(* FILE_LAST_MODIFIED_ON: 04/05/2006 AT: 09:00:25 *)
(*****
(* System Type : NetLinx *)
(*****
(* REV HISTORY: *)
(*****
(*
    $History: $
*)
DEFINE_DEVICE
dvTP = 10001:1:0

DEFINE_CONSTANT
// CONTACTS Listview button address
INTEGER btnListview = 11

DEFINE_VARIABLE

DEFINE_EVENT

    // The custom event that is raised whenever a contact on
    // the listview item is selected on the panel
CUSTOM_EVENT[dvTP,btnListview,LISTVIEW_ON_ROW_SELECT_EVENT]
{
    SLONG payloadId
    SLONG payloadType
    CHAR fields[3][16]
    CHAR name[DATA_MAX_VALUE_LENGTH]
    CHAR number[DATA_MAX_VALUE_LENGTH]
    CHAR image[DATA_MAX_VALUE_LENGTH]
    DATA_RECORD record

    // Get the data access ID from the custom event
    payloadId = custom.value1
    // Get the data type from the custom event
    payloadType = custom.value2

    if (payloadId > 0 && payloadType == DATA_STRUCTURE_DATA_RECORD)
    {
```

```
// Specify which fields we want to retrieve from the payload
fields[1] = 'column1'
fields[2] = 'column3'
fields[3] = 'column2'

// Populate a record with the requested fields from the event
if (DATA_GET_EVENT_RECORD(dvTP, payloadId, fields, record) > 0)
{
    // All is well so far so retrieve the values that we are
    // interested in from the selection that the user made on
    // the panel.
    name = record.content[1].value
    number = record.content[2].value
    image = record.content[3].value
    // Put the name and number that was selected on a popup and
    // show the popup
    SEND_COMMAND dvTP, "'^TXT-50,0,' ,name"
    SEND_COMMAND dvTP, "'^TXT-51,0,' ,number"
    SEND_COMMAND dvTP, "'^BMX-52,0,' ,image,' ,1,10'"
    SEND_COMMAND dvTP, "'^PPN-Calling'"
    SEND_COMMAND dvTP, "'^PPT-Calling;50'"
}
}

DEFINE_PROGRAM

(*****
( *                END OF PROGRAM                *)
( *          DO NOT PUT ANY CODE BELOW THIS COMMENT          *)
(*****)
```

2. Use NetLinx Studio 4 to compile the code (select **Build > Compile**).
3. Use NetLinx Studio 4 to transfer the AXS file to the NX Master:
  - a. Select **Tools > File Transfer** to open the *File Transfer* dialog.
  - b. In the *Send* tab, click the **Add** button. This opens the *Select Files for File Transfer* dialog.
  - c. In the *Other* tab, select **Non-System File** and click **Add**.
  - d. Select the compiled NetLinx code (in this example, "ISE\_CUSTOM\_EVENT.axs") and click **Open**. This opens the *Enter Device Mapping* dialog.
  - e. Review and edit the D:P:S settings for the target NX Master (leave the *Master Directory* field empty), and click **OK** to close the *Enter Device Mapping* dialog and return to the *Select Files for File Transfer* dialog.
  - f. Select **OK** to return to the *File Transfer* dialog.
  - g. In the *File Transfer* dialog, click **Send** to initiate the file transfer.
  - h. The progress of the transfer is indicated in the Output Bar.



NOTE

The custom event code shown above is included in the NetLinx Studio Workspace file (CSV.apw) that is in the Conference Rooms.ZIP file.



## 9) Transfer the TPDesign5 Project to the Touch Panel

At this point, everything is ready to go: the NX Master has the code to handle custom events and the TPD5 project file is handling the data source/mapping for the Listview button.

The only thing left to do is to transfer the TPD5 project containing the Listview button, data source reference and image references to the G5 touch panel:

1. In TPDesign5, select **Transfer > Connect** to open the *Connect* dialog (FIG. 46):

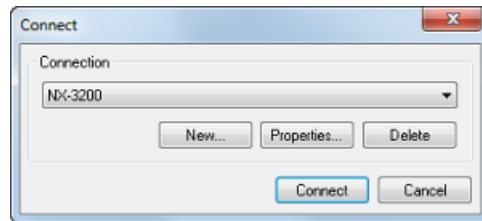


FIG. 46 Connect dialog



*If the Master has never been connected to before, a new connection will need to be configured. Refer to the File Transfer Operations section on page 261 for details.*

2. Select the connection configuration for the target NX Master from the *Connection* drop-down list, and click **Connect**.

Once a connection has been established with the Master, select **Transfer > Send to Panel** to open the *Send to Panel* dialog (FIG. 47):

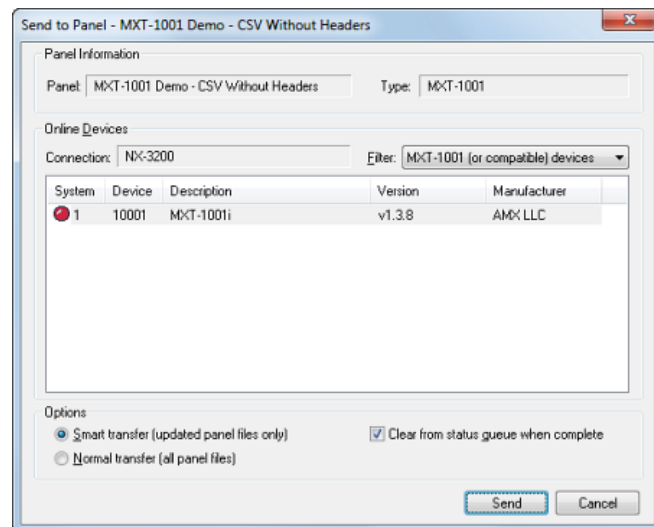


FIG. 47 Send To Panel dialog

3. Click **Send** to begin the file transfer.

When the transfer is complete, the Listview button should appear on the Page it was added to.

## Example 2 (CSV File - No Headers) - Results

FIG. 48 shows an example of a basic Listview button created by following these steps:

The figure illustrates the process of creating a Listview button from a CSV file. It includes the following components:

- conference.csv**: A snippet of a CSV file with columns A, B, and C. The data rows are:
 

	A	B	C
1	Alan Turing	TURING.jpg	469-285-2921
2	Albert Einstein	EINSTEIN.jpg	469-285-2922
3	Werner Heisenberg	HEISENBERG.jpg	469-285-2923
4	Disney	WALTDISNEY.jpg	469-285-2924
5	Dr Martin Luth	MARTINLUTHER.	469-285-2925
6	Eleanor Roose	ELEANOR.jpg	469-285-2926
7	Emerson	EMERSON.jpg	469-285-2927
- Resource Manager (Images tab)**: Shows room icon images imported into the TPD5 project, including CHURCHILL.jpg, DYLAN.jpg, EDWINHUBBLE.jpg, LINCOLN.jpg, and ELEANOR.jpg.
- Resource Manager (Dynamic Data Sources tab)**: Shows the *Conference Rooms* data source imported into the TPD5 project, with a URL of `http://19.38.90.42/conference.csv`.
- Dynamic Data Mappings - Listview**: A dialog showing the mapping of data from the CSV file to Listview components:
  - Primary Text: `[[= [column1]`
  - Secondary Text: `[[= [column3]`
  - Image: `[[= [column2]`
- Resulting Listview button**: A button displayed on the panel, showing a list of items with a search bar at the top. The items are:
  - Alan Turing (469-285-2921)
  - Albert Einstein (469-285-2922)
  - Werner Heisenberg (469-285-2923)
  - Disney (469-285-2924)
  - Dr Martin Luther King Jr (469-285-2925)
  - Eleanor Roosevelt (469-285-2926)

**FIG. 48** Example Listview button based on "conference.csv"

Using the *conference.csv* file as it's data source:

- It displays each room's name (based on the data in Column #1) as the *Primary Text* component.
- It displays each room's phone number (based on the data in Column #3) as the *Secondary Text* component.
- It displays each room's icon (based on the data in Column #2) as the *Image* component.



While the Listview button shown in this example uses only basic design characteristics, note that Listview buttons support most of the same display options as other button types, including Radiant/Gradient fills, Text Effects, Opacity, etc... Use these options to create eye-catching designs, just like for any other button type.

**Reference: "conference.csv" (CSV File Without Headers)**

Alan Turing, TURING.jpg, 469-285-2921  
 Albert Einstein, EINSTEIN.jpg, 469-285-2922  
 Werner Heisenberg, HEISENBERG.jpg, 469-285-2923  
 Disney, WALTDISNEY.jpg, 469-285-2924  
 Dr Martin Luther King Jr, MARTINLUTHER.jpg, 469-285-2925  
 Eleanor Roosevelt, ELEANOR.jpg, 469-285-2926  
 Emmeline Pankhurst, PANKHURST.jpg, 469-285-2927  
 Frank Sinatra, FRANKSINATRA.jpg, 469-285-2928  
 Henry Ford, HENRYFORD.jpg, 469-285-2929  
 Jackie Robinson, JACKIEROBINSON.jpg, 469-285-2930  
 Jean Piaget, PIAGET.jpg, 469-285-2931  
 Margaret Thatcher, MARGARETHATCHER.jpg, 469-285-2932  
 Muhammad Ali, MUHAMMADALI.jpg, 469-285-2933  
 Nelson Mandela, NELSONMANDELA.jpg, 469-285-2934  
 Rachel Carson, RACHELCARLSON.jpg, 469-285-2935  
 Scott Miller, MILLER.jpg, 469-285-2936  
 Sigmund Freud, SIGMUNDFREUD.jpg, 469-285-2937  
 Teddy Roosevelt, THEODOREROOSEVELT.jpg, 469-285-2938  
 The Beatles, THEBEATLES.jpg, 469-285-2939  
 TS Eliot, TSELIOT.jpg, 469-285-2940  
 Warren Buffet, WARRENBUFFET.jpg, 469-285-2941  
 Winston Churchill, CHURCHILL.jpg, 469-285-2942  
 Edwin Hubble, EDWINHUBBLE.jpg, 469-285-2943

**Conference Rooms Demo File ("Conference.ZIP")**

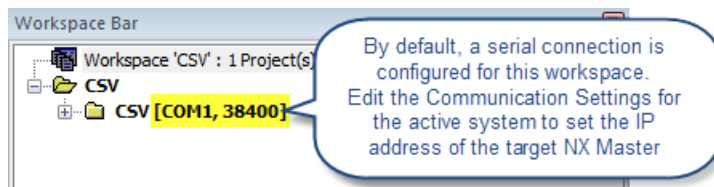
Demo (ZIP) files for the Listview examples presented here are available to download from the UI RESOURCE CENTER at [www.amx.com](http://www.amx.com). The preceding example followed the *Conference Rooms* demo. The Conference Rooms demo ZIP file (**Conference.ZIP**) contains the following:

Conference.ZIP Contents	
File	Description
<b>CSV.TP5</b>	TPDesign5 project file with the Listview button configured according to this example.
<b>"conference images" folder</b>	This folder contains the images used for the Listview button in this example.
<b>CSV.apw</b>	NetLinx Studio 4 Workspace file, with the Listview demo custom event defined. This Workspace contains the following files: <ul style="list-style-type: none"> <li>• CSV_CUSTOM_EVENT.axs</li> <li>• CSV_CUSTOM_EVENT.src</li> <li>• CSV_CUSTOM_EVENT.tkn</li> <li>• CSV_CUSTOM_EVENT.tko</li> </ul>
<b>conference.csv</b>	This CSV file (with no headers) will be used as the data source file for this example.

To use this demo:

1. Download the *Conference.ZIP* file and extract it's contents to a known location.
2. Launch TPDesign5 and open the *CSV.TP5* project file. Use TPDesign5 to set to the Host (IP) address for the data source file:
  - a. Open the Resource Manager to the *Dynamic Data Sources* tab, and double-click on the **conference.csv** file to access the *Edit Dynamic Data Source* dialog.
  - b. Edit the **Host** field with the IP address of the NX Master that will host the file. Click **OK** to save changes and close this dialog.
  - c. Close the Resource Manager.
  - d. Save changes and close the TP5 project.

3. Use NetLinx Studio 4 to transfer the *conference.csv* data source file to the target NX Master, so that it will be hosted on the Master:
  - a. Launch NetLinx Studio 4 and select **Tools > File Transfer** to open the *File Transfer* dialog (*Send* tab).
  - b. Click **Add** to open the *File Transfer* dialog, and open the **Other** tab.
  - c. Select *Non-System File* and click **Add** to access the *Open* dialog.
  - d. Locate and select the *channelList.csv* file and click **Open**
  - e. In the *Enter Device Mapping Information* dialog, review (and edit if necessary) the mapping information for this file, and click **OK** to return to the *Select File for File Transfer* dialog.
  - f. Click **OK** to return to the *File Transfer* dialog. Verify that the *conference.csv* file is selected for transfer, and click **Send**. Refer to 3) *Host a Data Source File (CSV without Headers) on the NX Master* on page 25 for more details.
4. In NetLinx Studio 4, open the *CSV.apw* workspace file (**File > Open Workspace**).  
This Workspace contains NetLinx source code that is pre-configured with a Custom Event for user selection, as well as a TPDesign5 project that includes a pre-configured Listview button that uses *conference.csv* as it's data source.
5. Build the Workspace: Select **Build > Build Active System**.
6. Transfer all files contained in the Workspace to the target NX Master:
  - a. Select **Settings > Active System Communication Settings** to open the *Communication Settings* dialog. Use the options in this dialog to establish a connection to the target NX Master. Note that by default, the workspace is configured to use Serial communication (FIG. 49):



**FIG. 49** NetLinx Studio 4 Workspace Bar - Conference demo (default communication settings)

This IP address should be the same as was specified for the data source file (see Step 3 above). See NetLinx Studio 4 online help for details on configuring communication settings.

- b. Select **Tools > File Transfer** to open the *File Transfer* dialog (*Send* tab). Remove any files (from previous transfer operations) that may be in the list.
- c. Click **Add** to open the *Select Files for File Transfer* dialog (*Current Workspace* tab).
- d. Click the top-level *Projects* directory to auto-select all files in the Workspace.
- e. Verify that the IP address indicated here indicates the correct NX Master, and click **OK** to save changes and return to the *File Transfer* dialog.
- f. In the *File Transfer* dialog, click **Send** to transfer the Workspace files to the target NX Master.

## Example 3: XML File/XPort Server

### Overview

The following instructions describe using the XML File/XPort Server demo for creating Listview buttons with a dynamic data source in the form of a Xport-generated XML file that is hosted on an NX Master.

For use as data source files for Listview buttons, XML files must be in the format that is exported by XPort servers - this represents the "AMX Standard" XML format.

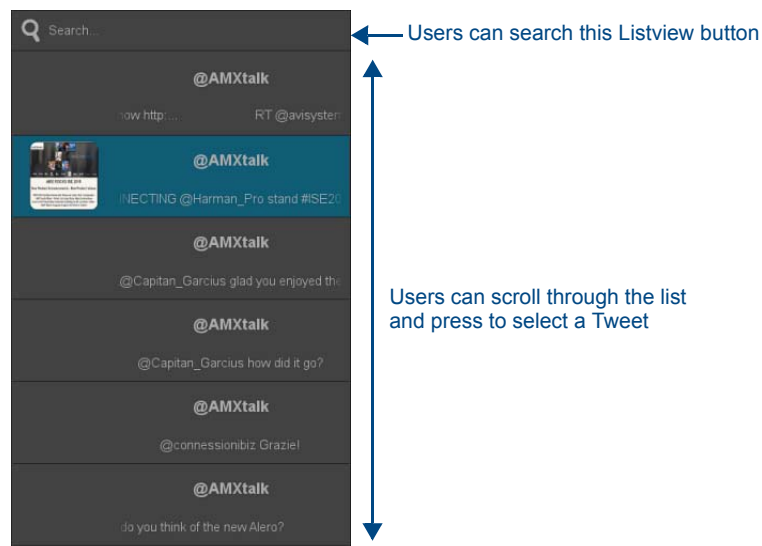


*Listview buttons do not support grouped data. If data is grouped, it cannot be displayed on a G5 Listview using the amxstandard.xml XPort output format.*



*This set of instructions uses files that are included in the "Twitter.ZIP" demo file which is available to download from the UI RESOURCE CENTER at [www.amx.com](http://www.amx.com).*

The resulting Listview button will display a listing of messages from the "AMX Talk" Twitter account, with the Twitter user name (@AMX Talk), the text of each message in the feed, and an image if one is associated with the message (FIG. 50):



**FIG. 50** Example - Listview button based on "amxstandard.xml" (AMX Talk Twitter account)

### Before You Begin

Download the *Twitter.ZIP* file from the UI RESOURCE CENTER at [www.amx.com](http://www.amx.com) and extract it's contents to a known location.

#### 1) Create Twitter Feed on the XPort Server

The XML file that will serve as the data source file in this example is generated by an XPort server. The following instructions describe generating an XML file for a Twitter feed from the XPort server:



*In order to create a new Twitter feed on the XPort server, a valid Twitter user account is required.*

Use a web browser to access the XPort server's configuration (Home) page: Enter the XPort server's IP address in the address bar, followed by "/xport/" (for example, "[10.35.90.45/xport/](http://10.35.90.45/xport/)").

1. In the XPort Server's Dashboard page under *External Feeds*, hover the cursor over the **Twitter** configuration icon to access the **[+] Create New Feed** option (FIG. 51):



FIG. 51 XPort Dashboard page > External Feeds > Twitter [+] Create New Feed

- Click on **[+] Create New Feed** to open the *Create a New Feed* page. This example will create a feed named "AMXTalk" for a Twitter user named "@amxtalk". Fill in the options on this page as shown in FIG. 52:

FIG. 52 Create New Feed page indicating the AMXTalk twitter feed

- In the *Name* field, enter "AMXTalk".
  - In the *Description* field, enter "amx".
  - Under *Feed Type*, select **User Tweets**.
  - Under *User Name*, enter "@amxtalk".
  - Under *Refresh Interval*, select **5 minutes**.
- Click **Authorize** to grant access to an active Twitter account via Twitter.com (FIG. 53):

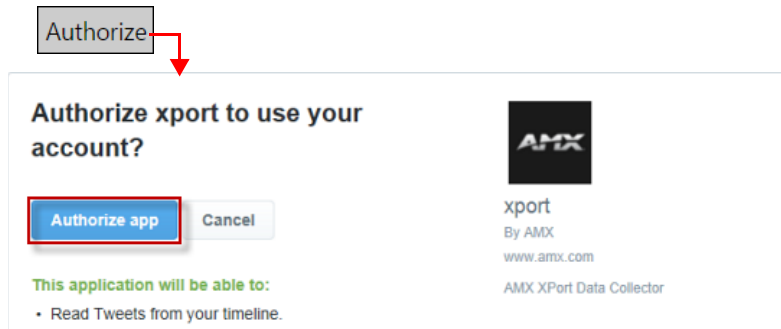


FIG. 53 Twitter.com - Authorize app page

4. In the *Create New Feed* page, click **Create Feed**. Xport will indicate that a new feed is being prepared (FIG. 54):

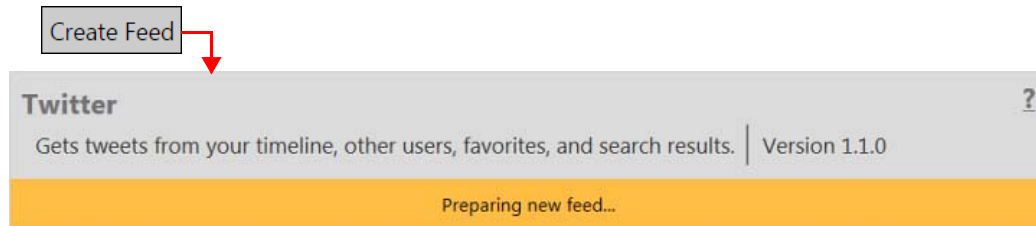


FIG. 54 XPort - Creating new Twitter feed

5. When the feed is ready, the Twitter feed page presents several **Output Feed** options (FIG. 55):

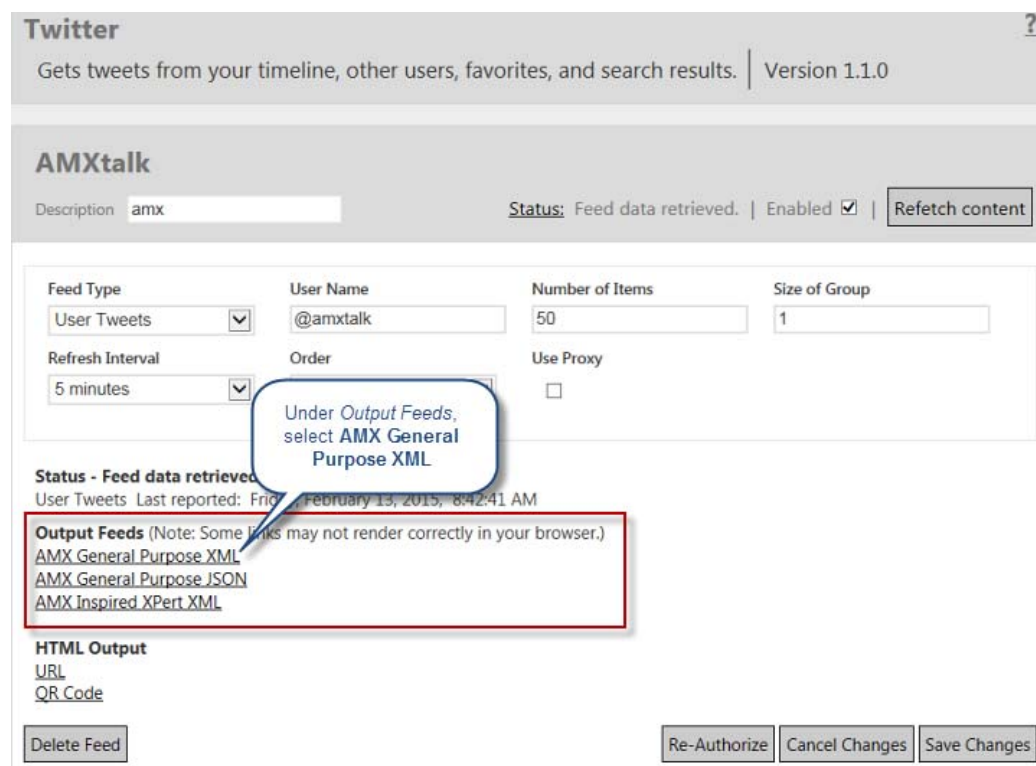


FIG. 55 XPort - Twitter (AMXTalk) feed page

## 2) Generate the "amxstandard.xml" file

In the Twitter (AMXTalk) feed page, under *Output Feeds*, click on the **AMX General Purpose XML** link to view the XML. The first few lines of the *amxstandard.xml* file are shown below (FIG. 56):

```
<?xml version="1.0" encoding="UTF-8" ?>
<data timestamp="Fri, 13 Feb 2015 10:52:43 GMT">
  <feed>
    <name>AMXTalk</name>
    <description>amx</description>
    <source/>
    <lastupdate>2015-02-13 10:52:43Z</lastupdate>
    <recordset count="1">
      <recordset id="AMXTalk">
        <records count="50">
          <record>
            <metadata>
              <field id="timestamp" type="datetime" format="ISO-8601" label="Record timestamp">2015-02-12 20:02:38Z</field>
              <field id="date created" type="date" format="ISO-8601" label="date created">2015-02-12</field>
            </metadata>
            <content>
              <field id="time created" type="time" format="ISO-8601" label="time created">8:02:38 PM</field>
              <field id="profile image" type="image" format="url" label="profile image">
                http://10.35.82.107/xport/feeds/twitter/amxtalk/cache_Fd01701i_normal.jpeg
              </field>
              <field id="photo" type="image" format="url" label="photo"/>
              <field id="user name" type="string" label="user name">AMX</field>
              <field id="screen name" type="string" label="screen name">@AMXTalk</field>
              <field id="text" type="string" label="text">
                RT @avisystems: Need a flexible platform for meeting rooms to share content? Check out the new Enzo from @AMXTalk at the #AVITechShow http://...
              </field>
            </content>
          </record>
          <record>
            <metadata>
              <field id="timestamp" type="datetime" format="ISO-8601" label="Record timestamp">2015-02-12 15:46:56Z</field>
              <field id="date created" type="date" format="ISO-8601" label="date created">2015-02-12</field>
            </metadata>
```

FIG. 56 AMX General Purpose XML



NOTE

*If using Internet Explorer, it is necessary to save the file, then open it (via **File > Open**) to view the XML.*

It will be necessary to understand the contents of the data source file in order to map the data to the Components in the Listview button later in this example.

Also, note the path indicated in the address bar of the browser when the XML is displayed. This address will be used later in this example to identify this file as a dynamic data source.

See page 52 to view this file.

## 3) Create (draw) a Listview button

1. In TPDesign5, open a Page and use the Button Draw tool to create a new button.
2. With the new button selected, click the **Type** (General) property and select **Listview** from the drop-down of button types. This selection sets the new button as a Listview button, and enables a set of Listview-specific properties (FIG. 57):

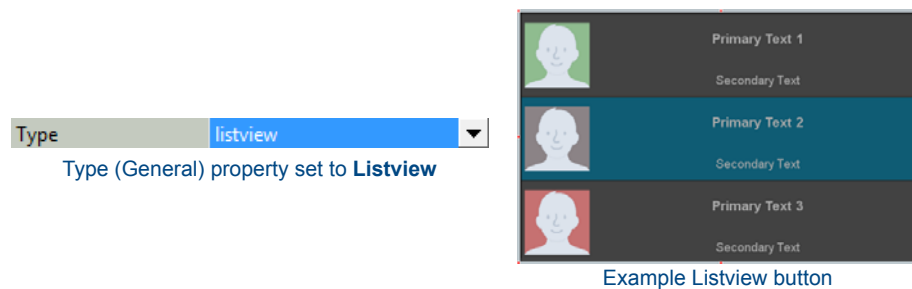


FIG. 57 Type (General) Property set to Listview



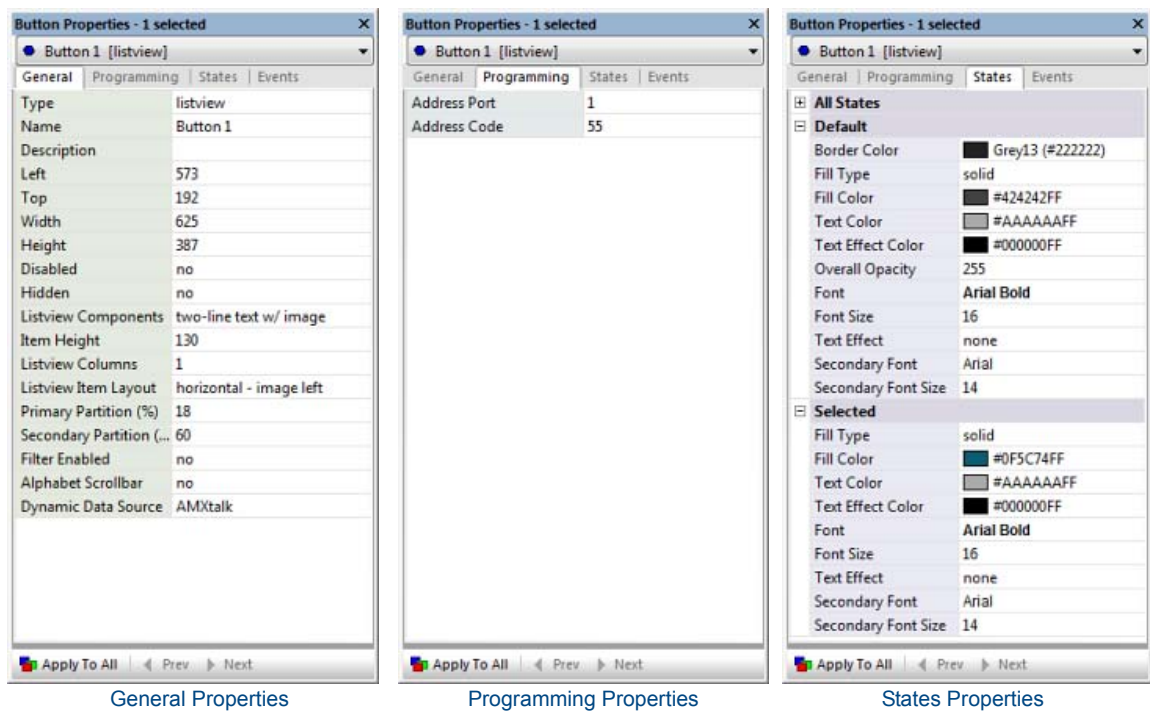
NOTE

*The "TWITTER.TP5" file included in the Twitter demo has a Listview button already drawn on the "Main" page.*



#### 4) Set Listview Button Properties

Use the options in the Properties window to view/edit the *General*, *Programming* and *States* properties for the Listview button. The settings used in this demo are shown in FIG. 58:



**FIG. 58** Properties for the Listview Button

Refer to the *Working With Listview Button Properties* section on page 3 for details on Listview-specific button properties.

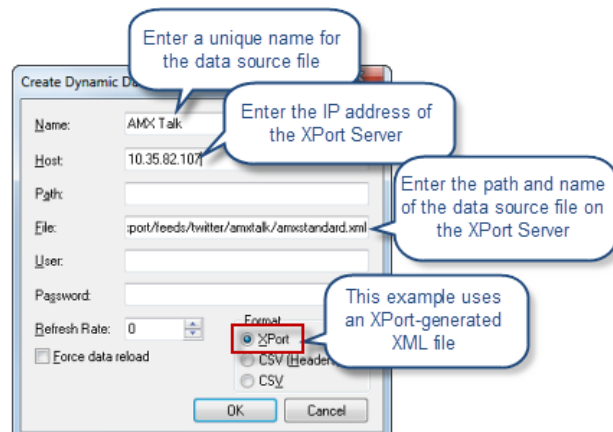


*The Listview button in the Twitter demo is pre-configured with the General, Programming and States properties shown above.*

#### 5) Add Dynamic Data Source to the Project

To add the data source file (amxstandard.xml) to the TPDesign5 project:

1. Open the Resource Manager to the *Dynamic Data Sources* tab and click **New** to open the *Create Dynamic Data Source* dialog (FIG. 59):



**FIG. 59** Create Dynamic Data Source dialog with Example data (amxstandard.xml)

2. In the *Name* field, enter a unique friendly name for the data source. For this example, enter "AMX Talk".

3. In the *File* field, enter a file name that indicates the full path to the location of the source file on the XPort Server. This information can be retrieved from the browser window, when the *amxstandard.xml* file is opened via the *AMX General Purpose XML* link on the Twitter configuration page on the XPort server (FIG. 60):

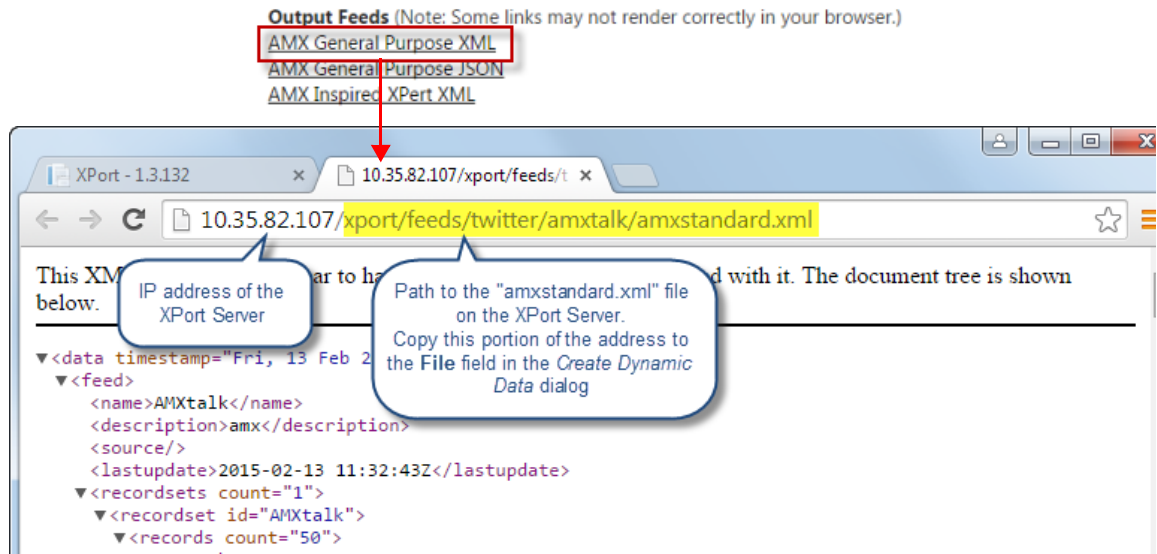


FIG. 60 Path to the "amxstandard.xml" file on the XPort Server

Note that the address/path displayed in the browser's address tab includes the IP address of the XPort Server (see "10.35.82.107" in the example above), and the path to the file ("xport/feeds/twitter/amxtalk/amxstandard.xml")

Enter the *IP address* in the **Host** field.

Enter the *path information* in the **File** field. Do not include the forward slash at the beginning of the path.

4. In the *User* field, enter the user name required by the NX Master or server for authentication (if required).
5. In the *Password* field, enter the password required by the NX Master or server for authentication (if required).
6. In the *Refresh Rate* field, use the up/down arrows to adjust the number of seconds between refreshes in which the resource is downloaded again. Refreshing resources will cause the button displaying that resource to refresh as well. The default value is 0, which means that the resource is only downloaded once.
7. Under *Format*, select **Xport**, since the data source file in this example (amxstandard.xml) uses an XPort-generated XML file.
8. Click **OK** to save changes and close this dialog. The new data source is indicated in the Resource Manager - Dynamic Data Sources tab (FIG. 61):

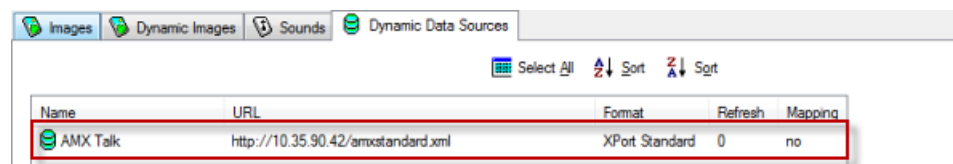


FIG. 61 Resource Manager - Dynamic Data Sources tab indicating "amxstandard.xml" as the data source



The *Listview* button in the *TWITTER.TP5* file is pre-configured to use *AMX Talk* (amxstandard.xml) as its data source file. However, it is necessary to update the *Host* address with the IP address of your NX Master as shown above. Double-click on *Conference Rooms* in the Resource Manager to open the *Edit Dynamic Data Source* dialog and update accordingly.

## 6) Map the Data from the Data Source File to the Listview Button Components

It is necessary to map the data in the *amxstandard.xml* file to the three fields that comprise the Listview button layout. These three fields (called Components in TPDesign5) are: *Primary Text*, *Secondary Text* and *Image* (FIG. 62):

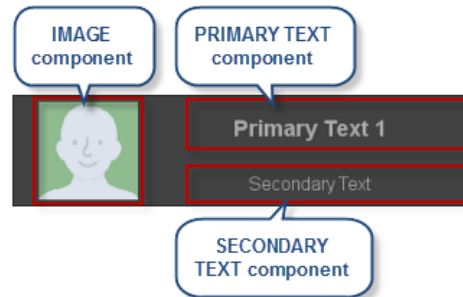


FIG. 62 Listview Button - Components

### Step One: Analyze the Data Source

It is necessary to understand the contents of the data source file in order to map the data to the Components in the Listview button. In this example, the *amxstandard.xml* file contains several IDs that will be mapped to the Listview button: *NAME*, *CHANNEL*, *ICON* and *RATING* (FIG. 63):

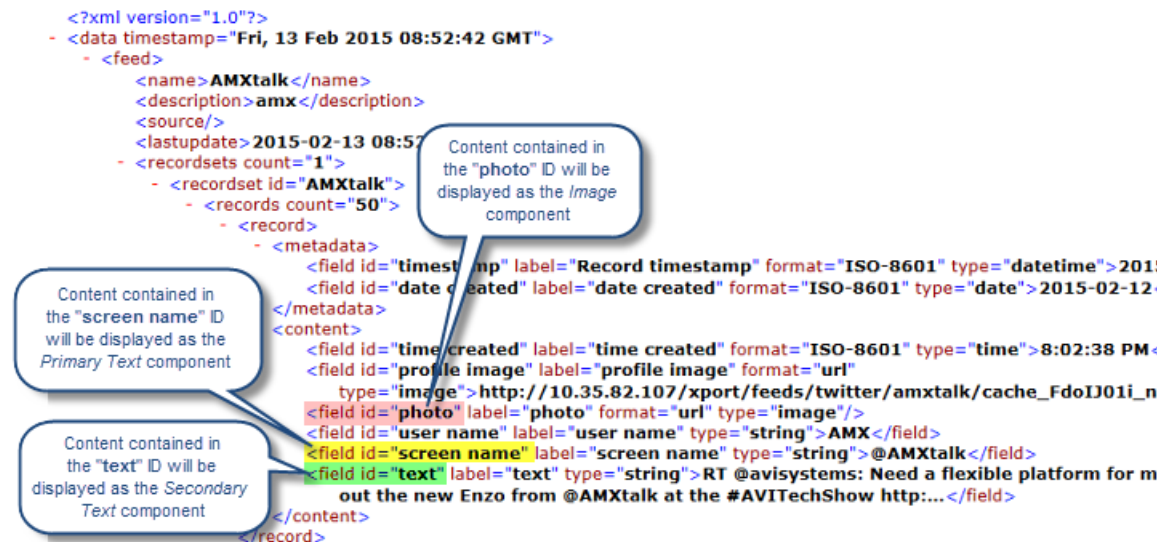


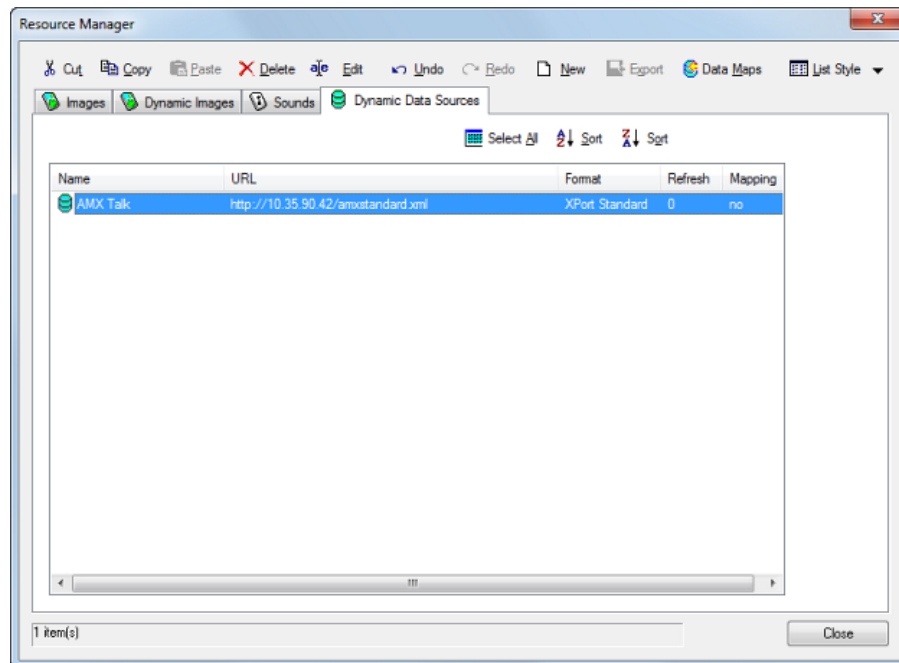
FIG. 63 Understanding the contents of the data source file - amxstandard.xml (Twitter feed)

In this example:

- The contents of the **"screen name"** ID will be mapped to display as the *Primary Text* component of the list items in the Listview button.
- The contents of the **"text"** ID will be mapped to display as the *Secondary Text* component of the list items in the Listview button.
- The contents of the **"photo"** ID will be mapped to display as the *Image* component of the list items in the Listview button.

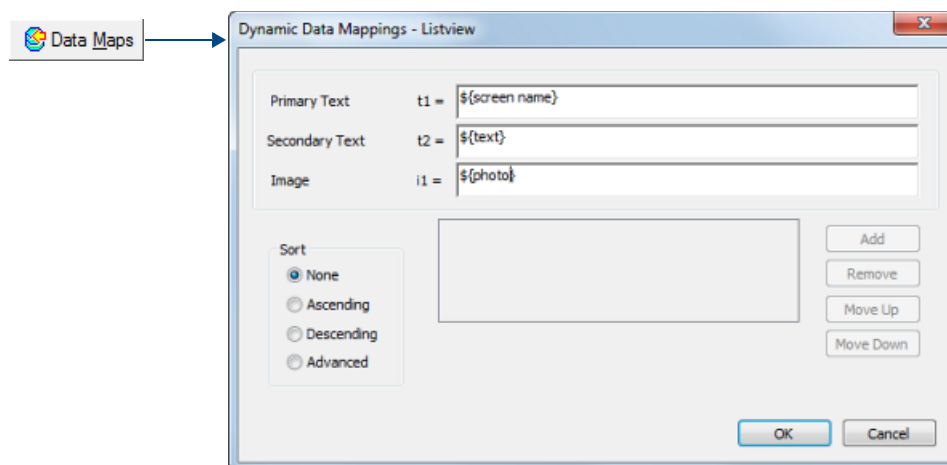
### Step Two: Map the Data to Components of the Listview button

1. With the Listview button selected, open the Resource Manager to the *Dynamic Data Sources* tab.
2. Select the data source (*AMX Talk*) that is assigned to the Listview button (as described on page 48):



**FIG. 64** Resource Manager - Dynamic Data Source tab

3. Click the **Data Maps** button to access the *Dynamic Data Mappings - Listview Buttons* dialog (FIG. 65):



**FIG. 65** Dynamic Data Mappings - Listview dialog (with example data indicated)

4. Use the fields in this dialog to specify the device mapping for the selected Listview button and the selected Data Source (see *Dynamic Data Mappings - Syntax Requirements (XML)* below).



*The Listview button in the Twitter demo is pre-configured with the data mapping settings shown above.*

### Dynamic Data Mappings - Syntax Requirements (XPort-Generated XML)

Note that the syntax requirements for these fields depends on the type of file used as the data source. The data source file in this example uses an XPort-generated XML file. The syntax requirements for data mapping to an XPort-generated XML file is described below:

Dynamic Data Mappings - Syntax Requirements (XPort-generated XML file)	
<b>Primary Text:</b> For XPort-generated XML files, the syntax is: <b>#{ID}</b> Following this syntax, enter the name of the ID in the data source file to be displayed as the Primary Text component of the Listview button. In this example, <i>amxstandard.xml</i> lists the name associated with the Twitter user in the "screen name" ID. To display the contents of the "screen name" ID as the Primary Text component, enter <b>#{screen name}</b> in the <i>Primary Text</i> field: <div> <div>Primary Text</div> <div>t1 = </div> <div>#{screen name}</div> </div>	
<b>Secondary Text:</b> For XPort-generated XML files, the syntax is: <b>#{ID}</b> Following this syntax, enter the name of the ID in the data source file to be displayed as the Secondary Text component of the Listview button. In this example, <i>amxstandard.xml</i> lists the caption associated with each "Tweet" in the "text" ID. To display the contents of the "text" ID as the Secondary Text component, enter <b>#{text}</b> in the <i>Secondary Text</i> field: <div> <div>Secondary Text</div> <div>t2 = </div> <div>#{text}</div> </div>	
<b>Image:</b> For XPort-generated XML files, the syntax is: <b>#{ID}</b> Following this syntax, enter the name of the ID in the data source file to be displayed as the Image component of the Listview button. In this example, <i>amxstandard.xml</i> lists the image associated with each "Tweet" in the "photo" ID. To display the contents of the "photo" ID as the Secondary Text component, enter <b>#{photo}</b> in the <i>Image</i> field: <div> <div>Image</div> <div>i1 = </div> <div>#{photo}</div> </div>	



NOTE

The fields in the Dynamic Data Mappings - Listview Buttons dialog are case-sensitive.

## 7) Assign a Data Source file to the Listview Button

The data source (amxstandard.xml) is associated with the Listview button via the *Dynamic Data Source* property (in the *General* tab of the Properties window):

1. With the Listview button selected, click the browse button in the **Dynamic Data Source** (General) property to open the *Select Resource* dialog (FIG. 66):

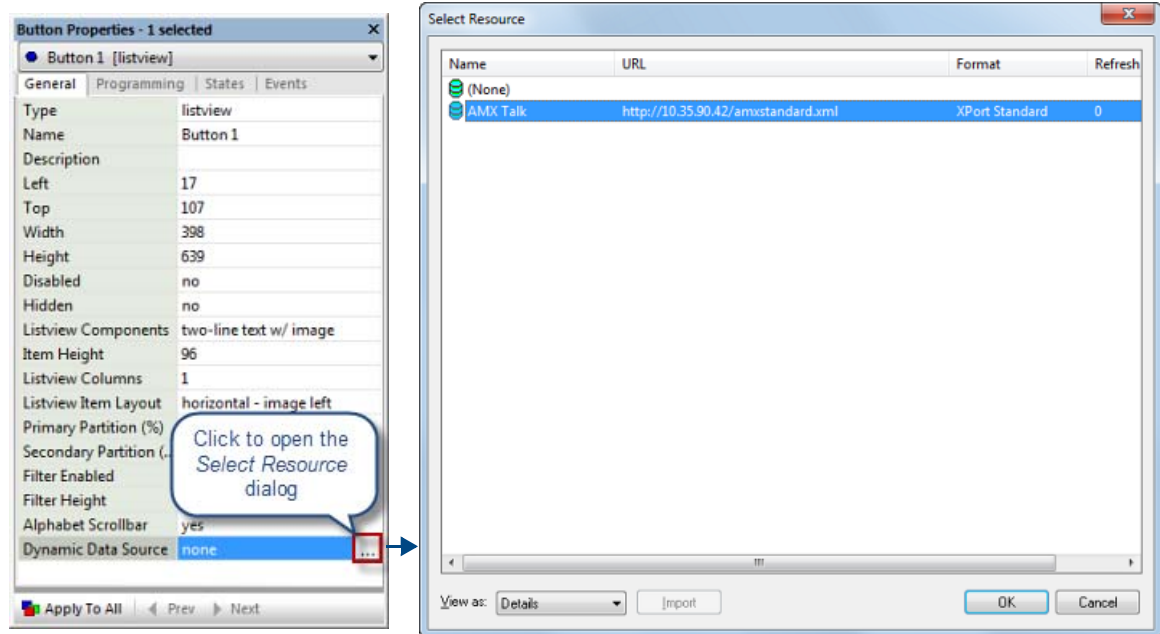


FIG. 66 Dynamic Data Source (General) Property and Select Resource dialog

2. Select the XML file to use as the data source (in this example, "AMX Talk").
3. Click **OK** to close this dialog.
4. The selected Data Source file is indicated in the *Dynamic Data Source* property (see "AMX Talk" in FIG. 67):

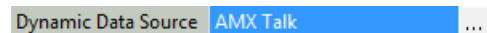


FIG. 67 Dynamic Data Source property indicating "AMX Talk"



NOTE

The "TWITTER.TP5" file included in the Twitter demo has "AMX Talk" already assigned as the Dynamic Data Source for the Listview button.

## 8) Write a Custom Event To Respond To User Selection

When the user selects an item on the Listview button, the entire record for that selection is sent to the NX Master. A `CUSTOM_EVENT` is raised and within this function the desired information can be retrieved for the selection. In this example, a popup window that displays the text and image for each message.

Listview buttons use the custom event parameter "LISTVIEW\_ON\_ROW\_SELECT\_EVENT" to provide the ability to configure a response to the selection of a list item in a Listview button in NetLinx code.

This custom event must be added to the NetLinx code on the NX Master.

1. Use NetLinx Studio 4 to add the following code to the `CUSTOM EVENT` section of the NetLinx program loaded on the Master:

```

PROGRAM_NAME='ISE_CUSTOM_EVENT'
(*****)
(*****)
(* FILE_LAST_MODIFIED_ON: 04/05/2006 AT: 09:00:25 *)
(*****)
(* System Type : NetLinx *)
(*****)
(* REV HISTORY: *)
(*****)
(*
    $History: $
*)

DEFINE_DEVICE
dvTP = 10001:1:0

DEFINE_CONSTANT

// Twitter Listview button address
INTEGER btnTWTLlistview = 55

DEFINE_VARIABLE

DEFINE_EVENT

// CUSTOM EVENT RAISED WHEN ITEM IN
// TWITTER LISTVIEW WIDGET IS SELECTED
CUSTOM_EVENT[dvTP,btnTWTLlistview,LISTVIEW_ON_ROW_SELECT_EVENT]
{
    SLONG payloadId
    SLONG payloadType
    CHAR fields[2][16]
    CHAR screenName[DATA_MAX_VALUE_LENGTH]
    CHAR text[DATA_MAX_VALUE_LENGTH]
    DATA_RECORD record

    // Get the data access ID from the custom event
    payloadId = custom.value1
    // Get the data type from the custom event
    payloadType = custom.value2

    if (payloadId > 0 && payloadType == DATA_STRUCTURE_DATARECORD)
    {
        // Specify which fields we want to retrieve from the payload
        fields[1] = 'screen name'
        fields[2] = 'text'
        // Populate a record with the requested fields from the event
        if (DATA_GET_EVENT_RECORD(dvTP, payloadId, fields, record) > 0)
        {
            // All is well so far so retrieve the values that we are
            // interested in from the selection that the user made on
            // the panel.
            screenName = record.content[1].value
            text = record.content[2].value

            // Put the name and number that was selected on a popup and
            // show the popup
            SEND_COMMAND dvTP, "'^TXT-56,0,'",screenName"
            SEND_COMMAND dvTP, "'^TXT-57,0,'",text"
            SEND_COMMAND dvTP, "'^PPN-Twitter'"
            SEND_COMMAND dvTP, "'^PPT-Twitter;50'"
        }
    }
}

DEFINE_PROGRAM
(*****)
(*                END OF PROGRAM                *)
(*                DO NOT PUT ANY CODE BELOW THIS COMMENT                *)
(*****)

```

2. Use NetLinx Studio 4 to compile the code (select **Build > Compile**).
3. Use NetLinx Studio 4 to transfer the AXS file to the NX Master:
  - a. Select **Tools > File Transfer** to open the *File Transfer* dialog.
  - b. In the *Send* tab, click the **Add** button. This opens the *Select Files for File Transfer* dialog.
  - c. In the *Other* tab, select **Non-System File** and click **Add**.

- d. Select the compiled NetLinx code (in this example, "ISE\_CUSTOM\_EVENT.axs") and click **Open**. This opens the *Enter Device Mapping* dialog.
- e. Review and edit the D:P:S settings for the target NX Master (leave the *Master Directory* field empty), and click **OK** to close the *Enter Device Mapping* dialog and return to the *Select Files for File Transfer* dialog.
- f. Select **OK** to return to the *File Transfer* dialog.
- g. In the *File Transfer* dialog, click **Send** to initiate the file transfer.
- h. The progress of the transfer is indicated in the Output Bar.



NOTE

The custom event code shown above is included in the NetLinx Studio Workspace file (TWITTER.apw) that is in the Twitter.ZIP file.

## 9) Transfer the TPD5 Project to the Touch Panel

At this point, everything is ready to go: the NX Master has the code to handle custom events and the TPD5 project file is handling the data source/mapping for the Listview button.

The only thing left to do is to transfer the TPD5 project containing the Listview button, data source reference and image references to the G5 touch panel:

1. In TPD5, select **Transfer > Connect** to open the *Connect* dialog (FIG. 68):

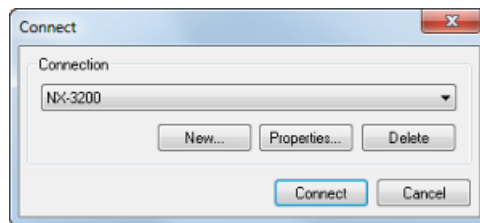


FIG. 68 Connect dialog



NOTE

If the Master has never been connected to before, a new connection will need to be configured. Refer to the *File Transfer Operations* section on page 261 for details.

2. Select the connection configuration for the target NX Master from the *Connection* drop-down list, and click **Connect**.

Once a connection has been established with the Master, select **Transfer > Send to Panel** to open the *Send to Panel* dialog (FIG. 69):

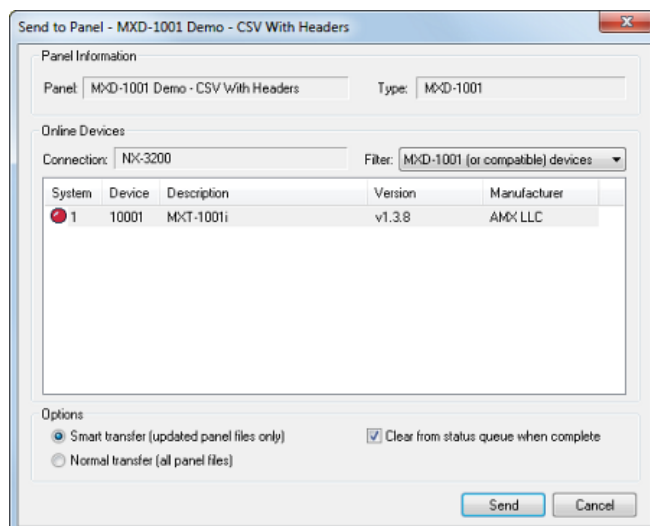


FIG. 69 Send To Panel dialog

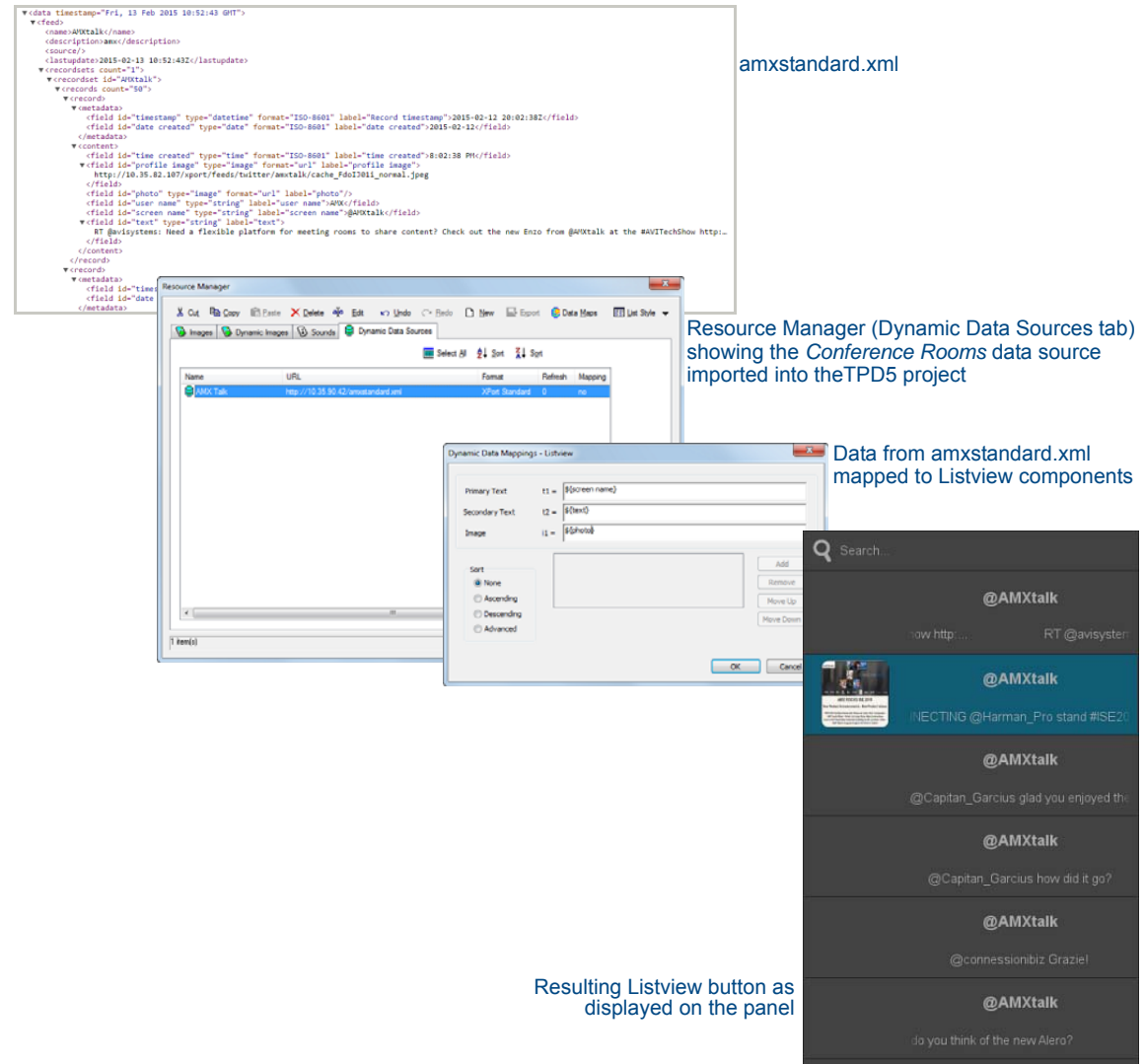


### 3. Click **Send** to begin the file transfer.

When the transfer is complete, the Listview button should appear on the Page it was added to.

### Example 3 (XML File/XPort Server) - Results

FIG. 70 shows an example of a basic Listview button created by following these steps:



**FIG. 70** Example Listview button based on "AMXTalk"

Using the *amxstandard.xml* file as it's data source:

- It displays the contents of the "screen name" ID as the *Primary Text* component.
- It displays the contents of the "text" ID as the *Secondary Text* component.
- It displays the contents of the "photo" ID as the *Image* component.

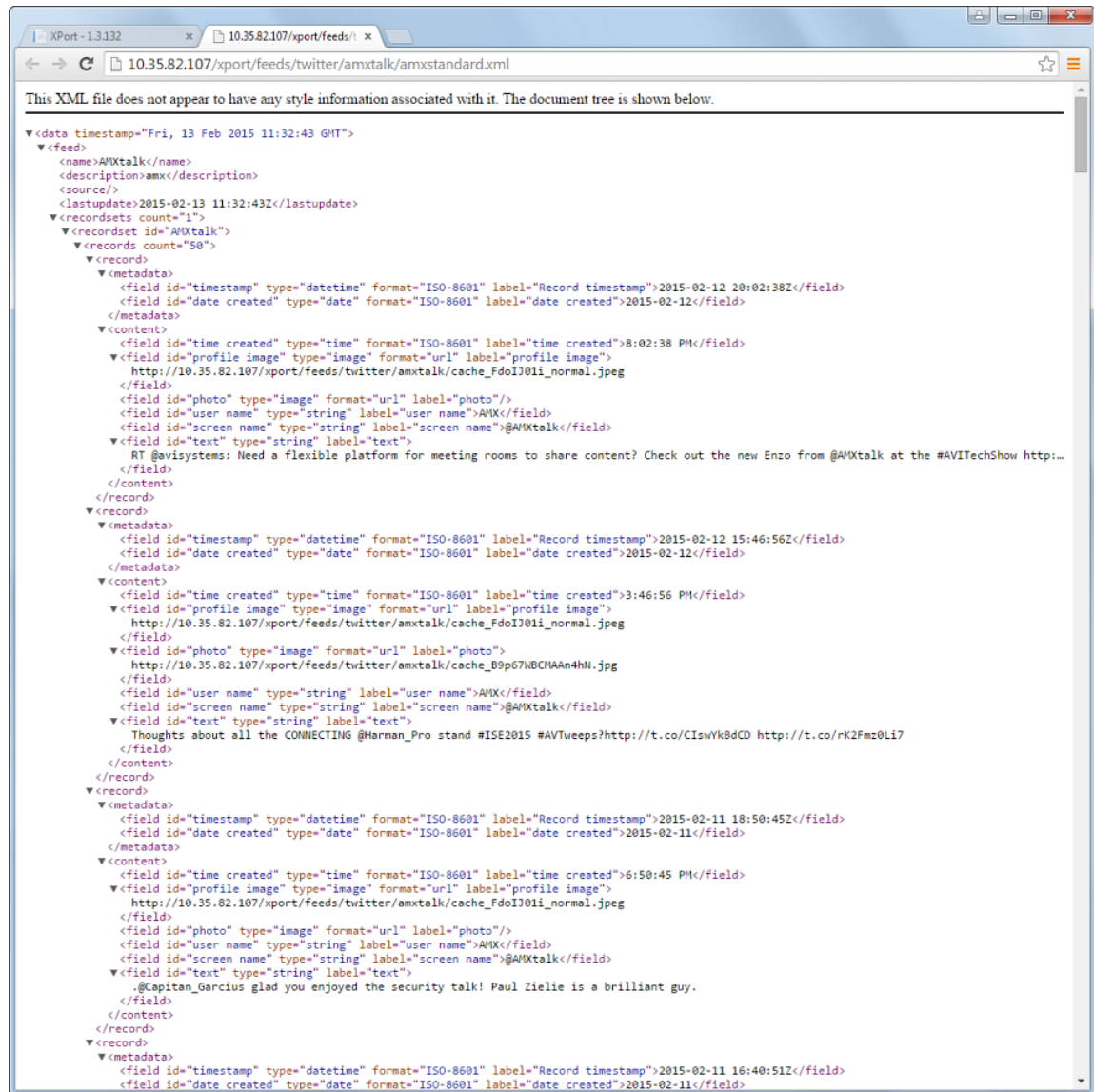


**NOTE**

While the Listview button shown in this example uses only basic design characteristics, note that Listview buttons support most of the same display options as other button types, including Radiant/Gradient fills, Text Effects, Opacity, etc... Use these options to create eye-catching designs, just like for any other button type.

**Reference: "amxstandard.xml"**

FIG. 71 provides a partial view of the "amxstandard.xml" file generated by the XPort server as "AMX General Purpose XML":



**FIG. 71** amxstandard.xml file (Generated by an XPort server) - partial view

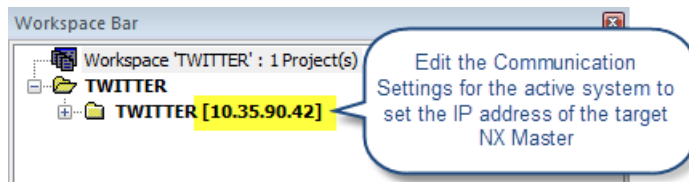
**Twitter (XPort XML) Demo File ("Twitter.ZIP")**

Demo (ZIP) files for the Listview examples presented here are available to download from the UI RESOURCE CENTER at [www.amx.com](http://www.amx.com). The preceding example followed the *Twitter* demo. The Twitter demo ZIP file (**Twitter.ZIP**) contains the following:

Twitter.ZIP Contents	
File	Description
<b>TWITTER.TP5</b>	TPDesign5 project file that includes a Listview button pre-configured to use the layout properties and data source file shown in the <i>Listview Button/Dynamic Data Example 3: XML File/XPort Server</i> example.
<b>TWITTER.png</b>	This image file is used as the Image component for the Listview button.
<b>TWITTER.apw</b>	NetLinx Studio 4 Workspace file, with the Listview demo custom event defined. This Workspace contains the TWITTER_CUSTOM_EVENT.axs file.

To use this demo:

1. Download the *TWITTER.ZIP* file and extract it's contents to a known location.
2. Launch TPDesign5 and open the *TWITTER.TP5* project file. Use TPDesign5 to set to the Host (IP) address for the data source file:
  - a. Open the Resource Manager to the *Dynamic Data Sources* tab, and double-click on **AMXTalk** to access the *Edit Dynamic Data Source* dialog.
  - b. Edit the **Host** field with the IP address of the XPort server that will provide the data. Click **OK** to save changes and close this dialog.
  - c. Close the Resource Manager.
  - d. Save changes and close the TP5 project.
3. In NetLinX Studio 4, open the *TWITTER.apw* workspace file (**File > Open Workspace**).  
This Workspace contains NetLinX source code that is pre-configured with a Custom Event for user selection, as well as a TPDesign5 project that includes a pre-configured Listview button that uses *AMXTalk* as it's data source.
4. Build the Workspace: Select **Build > Build Active System**.
5. Transfer all files contained in the Workspace to the target NX Master:
  - a. Select **Settings > Active System Communication Settings** to open the *Communication Settings* dialog. Use the options in this dialog to establish a connection to the target NX Master (FIG. 72):



**FIG. 72** NetLinX Studio 4 Select Files for File Transfer dialog (Current Workspace tab) - Projects directory selected

See NetLinX Studio 4 online help for details on configuring communication settings.

- b. Select **Tools > File Transfer** to open the *File Transfer* dialog (*Send* tab). Remove any files (from previous transfer operations) that may be in the list.
- c. Click **Add** to open the *Select Files for File Transfer* dialog (*Current Workspace* tab).
- d. Click the top-level *Projects* directory to auto-select all files in the Workspace.
- e. Verify that the IP address indicated here indicates the correct NX Master, and click **OK** to save changes and return to the *File Transfer* dialog.
- f. In the *File Transfer* dialog, click **Send** to transfer the Workspace files to the target NX Master.



# Example 4: NetLinX Data Source

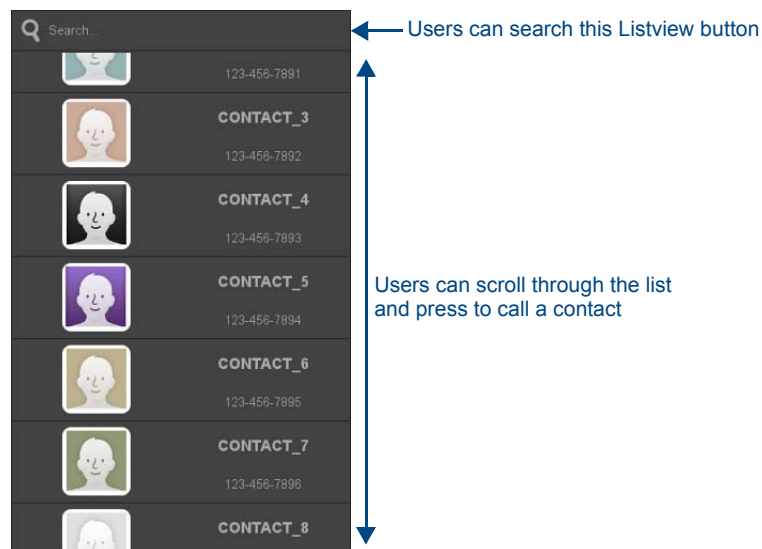
## Overview

The following section describes an example workflow for implementing a Listview button that uses NetLinX code as the data source. The use case for this example is that of a contact list for a SIP phone system. In this case, the user finds and selects a contact on the screen and then presses a call button to initiate the call. This is an example workflow for creating a Listview widget on a touchpanel page, creating a data source in NetLinX, configuring and populating the Listview and responding to a user selection.



*This set of instructions uses files that are included in the "NetLinXAPI.ZIP" demo file which is available to download from the UI RESOURCE CENTER at [www.amx.com](http://www.amx.com).*

The resulting Listview button will display a listing of phone contacts with each contact's name and phone number (FIG. 73):



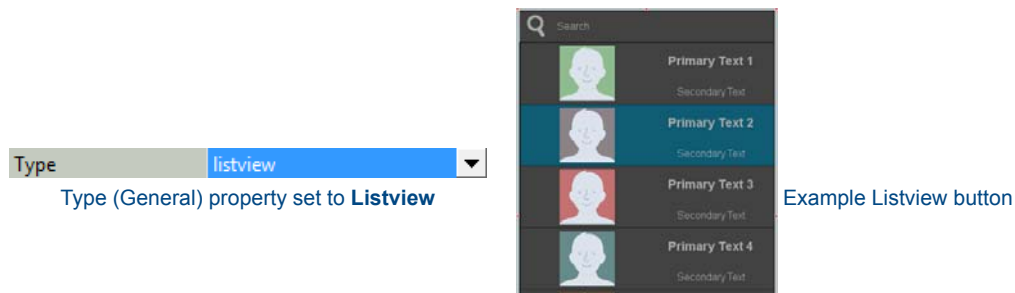
**FIG. 73** Example - Listview button based on "NetLinXAPI.axs"

## Before You Begin

Download the *NetLinXAPI.ZIP* file from the UI RESOURCE CENTER at [www.amx.com](http://www.amx.com) and extract it's contents to a known location.

### 1) Create (draw) a Listview button

1. In TPDesign5, open a Page and use the Button Draw tool to create a new button.
2. With the new button selected, click the **Type** (General) property and select **Listview** from the drop-down of button types. This selection sets the new button as a Listview button, and enables a set of Listview-specific properties (FIG. 74):



**FIG. 74** Type (General) Property set to Listview



NOTE

The "NetLinXAXI.TP5" file included in the NetLinXAXI demo has a Listview button already drawn on the "Main" page.

## 2) Set Listview Button Properties

Use the options in the Properties window to view/edit the *General*, *Programming* and *States* properties for the Listview button. The settings used in this demo are shown in FIG. 75:

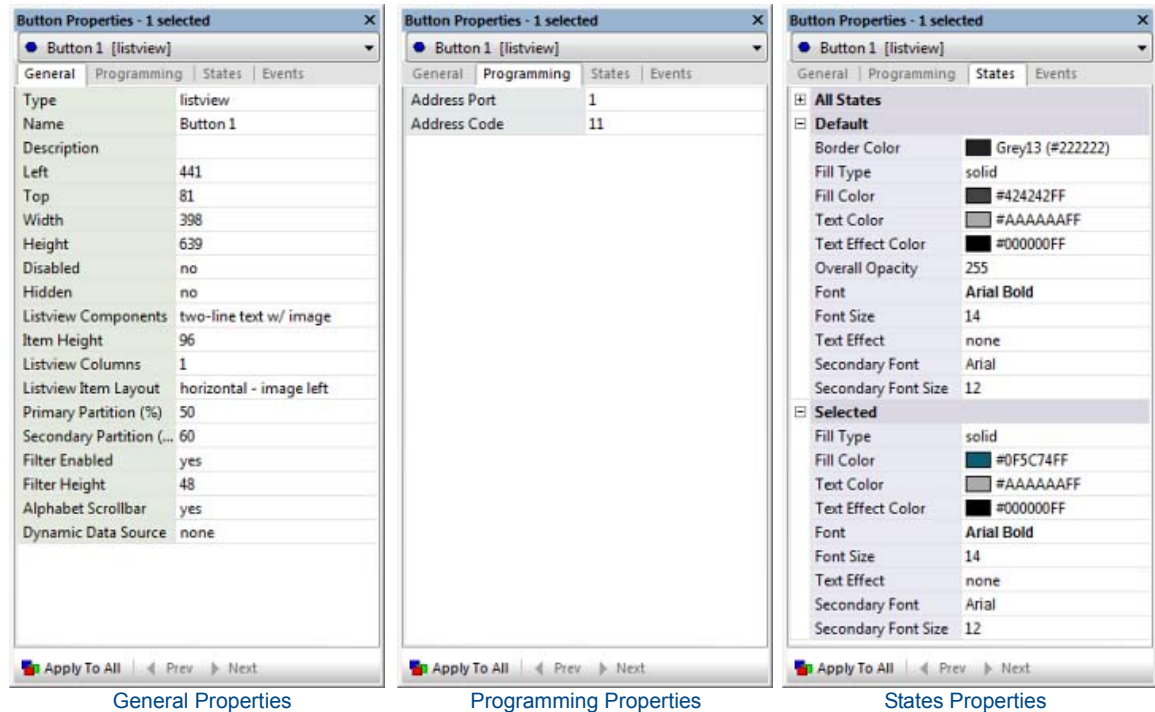


FIG. 75 Properties for the Listview Button



NOTE

The Listview button in the NetLinXAXI demo is pre-configured with the General, Programming and States properties shown above.

Refer to the *Working With Listview Button Properties* section on page 3 for details on Listview-specific button properties.

## 3) Create the Data Source

Follow the example *NetLinX Usage Example - ASCII* (below) to create a data source in NetLinX and publish the data source to the NX Master's internal web server.

The "Data\_PublishFeed()" function (see *NetLinX.axi*) will return a URL for the published data.

## 4) Configuring the Response to a User Selection

Follow the CUSTOM\_EVENT example at the end of the example below to retrieve the phone number that was selected by the user.

### NetLinX Usage Example - ASCII

Review the following code and read all comments to see how this file works:

```

PROGRAM_NAME='Listview Example'

DEFINE_DEVICE
dvTP = 10001:1:0

DEFINE_CONSTANT
// listview button address
INTEGER btnListview = 11

DEFINE_VARIABLE
//just a variable to hold our "published URL" value
CHAR publishedURL[DATA_MAX_VALUE_LENGTH]
//just a variable to hold our recordset ID
CHAR recordsetID[DATA_MAX_ID_LENGTH]

DEFINE_FUNCTION GenerateDataFeed()
{
    //we can't add fields to a record or a record to
    // a feed if they don't exist so lets define them here

    //DATA_FEED is a predefined structure in the NetLinX AXI
    //datafeed is our variable representing that structure
    STACK_VAR DATA_FEED datafeed

    //DATA_RECORD is a predefined structure in the NetLinX AXI
    //record is our variable representing that structure
    STACK_VAR DATA_RECORD record

    // -----
    // CREATE A NEW DATA FEED
    // -----
    //set the characteristics of the dataFeed (these are just descriptive strings)
    //name/description/source are defined in the DATA_FEED structure
    datafeed.name = 'phonelist'
    datafeed.description = 'Some Harman Employees'
    datafeed.source = 'NetLinX PhoneList'

    //we've defined all the values for our DATA_FEED
    //now we need to "create" our DATA_FEED
    DATA_CREATE_FEED(datafeed)

    // A recordset id is required for adding records to the feed
    recordsetID = 'recordSetPhoneList'

    // -----
    // DEFINE AND POPULATE THE DATA FIELDS
    // This example will have 10 names in a phone list
    // -----
    // Records can have metadata fields and content fields. In this
    // example we won't use any metadata
    SET_LENGTH_ARRAY(record.metadata, 1)
    // We will have 3 content fields per record: photo, name and phone number
    SET_LENGTH_ARRAY(record.content, 3)

    // Initialize the field attributes that will be the same for every record
    // the first field in a record will be the image
    record.content[1].id = 'photo';
    record.content[1].type = DATA_TYPE_IMAGE;
    record.content[1].format = DATA_FORMAT_URL;
    // When mapping the data to a listview widget on the panel and when using
    // the data in the custom event, the id field is used
    // The label can be something different from the id but in our case we'll
    // keep them the same
    record.content[1].label = 'photo';

```

```

// The second field in a record will be the name
record.content[2].id = 'name';
record.content[2].type = DATA_TYPE_STRING;
record.content[2].format = '';
record.content[2].label = 'name';
// The third field will be the phone number
record.content[3].id = 'number';
record.content[3].type = DATA_TYPE_STRING;
record.content[3].format = DATA_FORMAT_PHONE;
record.content[3].label = 'number';
// The next step is to put in the actual values for the 3 fields
record.content[1].value = 'http://server-lin/ftp/listview/CONTACT_1.png'
record.content[2].value = 'CONTACT_1'
record.content[3].value = '123-456-7890'
// Add the record to the feed
DATA_ADD_RECORD(datafeed.name, recordsetID, record)
// The same record can be reused for the rest of the list
// Just change the relevant values and add the record to the feed
record.content[1].value = 'http://server-lin/ftp/listview/CONTACT_2.png'
record.content[2].value = 'CONTACT_2'
record.content[3].value = '123-456-7891'
DATA_ADD_RECORD(datafeed.name, recordsetID, record)

record.content[1].value = 'http://server-lin/ftp/listview/CONTACT_3.png'
record.content[2].value = 'CONTACT_3'
record.content[3].value = '123-456-7892'
DATA_ADD_RECORD(datafeed.name, recordsetID, record)

record.content[1].value = 'http://server-lin/ftp/listview/CONTACT_4.png'
record.content[2].value = 'CONTACT_4'
record.content[3].value = '123-456-7893'
DATA_ADD_RECORD(datafeed.name, recordsetID, record)

record.content[1].value = 'http://server-lin/ftp/listview/CONTACT_5.png'
record.content[2].value = 'CONTACT_5'
record.content[3].value = '123-456-7894'
DATA_ADD_RECORD(datafeed.name, recordsetID, record)

record.content[1].value = 'http://server-lin/ftp/listview/CONTACT_6.png'
record.content[2].value = 'CONTACT_6'
record.content[3].value = '123-456-7895'
DATA_ADD_RECORD(datafeed.name, recordsetID, record)

record.content[1].value = 'http://server-lin/ftp/listview/CONTACT_7.png'
record.content[2].value = 'CONTACT_7'
record.content[3].value = '123-456-7896'
DATA_ADD_RECORD(datafeed.name, recordsetID, record)

record.content[1].value = 'http://server-lin/ftp/listview/CONTACT_8.png'
record.content[2].value = 'CONTACT_8'
record.content[3].value = '123-456-7897'
DATA_ADD_RECORD(datafeed.name, recordsetID, record)

record.content[1].value = 'http://server-lin/ftp/listview/CONTACT_9.png'
record.content[2].value = 'CONTACT_9'
record.content[3].value = '123-456-7898'
DATA_ADD_RECORD(datafeed.name, recordsetID, record)

record.content[1].value = 'http://server-lin/ftp/listview/CONTACT_10.png'
record.content[2].value = 'CONTACT_10'
record.content[3].value = '123-456-7899'
DATA_ADD_RECORD(datafeed.name, recordsetID, record)

// The final step is to publish the feed
publishedURL = DATA_PUBLISH_FEED(datafeed.name)
}

```



```

DEFINE_START
    GenerateDataFeed()

DEFINE_EVENT
DATA_EVENT[dvTP]
{
    ONLINE:
    {
        // Set the URL for the data source for the listviewer in the panel
        SEND_COMMAND dvTP, "^LVD-", ITOA(btnListView), ',', publishedURL"
        // Map the fields in the listviewer to the columns
        SEND_COMMAND dvTP, "^LVM-", ITOA(btnListView), ',,il=${photo}|t1=${name}|t2=${number}'"
        // Sort ascending by name
        SEND_COMMAND dvTP, "^LVS-", ITOA(btnListView), ',,${name};a"'
        // Command the listview to load the data from the master
        SEND_COMMAND dvTP, "^LVR-", ITOA(btnListView)"
    }
}

// The custom event that is raised whenever a listview item is selected on the panel
CUSTOM_EVENT[dvTP,btnListView,LISTVIEW_ON_ROW_SELECT_EVENT]
{
    SLONG payloadId
    SLONG payloadType
    //just a char array to hold the data we want to use in the custom event.
    CHAR fields[2][16]
    //char variables to hold our data for "name" & "number"
    CHAR name[DATA_MAX_VALUE_LENGTH]
    CHAR number[DATA_MAX_VALUE_LENGTH]

    //variable record, of type DATA_RECORD, to hold the record we retrieve from the custom event
    DATA_RECORD record
    // Get the data access ID from the custom event
    // variable is payloadID - custom.value1 is predefined
    payloadId = custom.value1
    // Get the data type from the custom event
    // variable is payloadType - custom.value2 is predefined
    payloadType = custom.value2

    if (payloadId > 0 && payloadType == DATA_STRUCTURE_DATARECORD)
    {
        // Specify which fields we want to retrieve from the payload
        // (these are the IDs we defined earlier)
        fields[1] = 'name'
        fields[2] = 'number'
        // Retrieve the record and get our requested fields
        if (DATA_GET_EVENT_RECORD(dvTP, payloadId, fields, record) > 0)
        {
            // The record existed and contained our fields
            // let's retrieve the values that we are interested in
            name = record.content[1].value
            number = record.content[2].value
            // Send the name & number that was retrieved to the appropriate buttons & show the popup
            SEND_COMMAND dvTP, "^TXT-50,0,',name"
            SEND_COMMAND dvTP, "^TXT-51,0,',number"
            SEND_COMMAND dvTP, "^PPN-Calling'"
        }
    }
}

}

(*****
*)          THE ACTUAL PROGRAM GOES BELOW          (*)
*****)
DEFINE_PROGRAM
(*****
*)          END OF PROGRAM          (*)
*)          DO NOT PUT ANY CODE BELOW THIS COMMENT          (*)
*****)

```



The NetLinx code shown above is included in the NetLinx Studio Workspace file (*NetLinxAPI.apw*) that is in the *NetLinxAPI.ZIP* file.

Update this code as necessary to reference your NX Master.

In order for this code to work with your Master, all instances of

`'http://server-lin/ftp/listview/CONTACT_1.png'`

must be updated to indicate the IP address of your NX Master.

For example, a Master with the IP address of "10/35.90.42" would require the following update

`'http://10.35.90.42/CONTACT_1.png'`

## 5) Compile the Code

In NetLinx Studio 4, select **Build > Build Active System** to compile the NetLinx code.

## 6) Transfer the Workspace to the NX Master

Use NetLinx Studio 4 to transfer the NetLinx code (*NetLinxAPI.tkn* and *NetLinxAPI.src*) files as well as the TPDesign5 project file (*NetLinxAPI.TP5*):

1. Select **Tools > File Transfer** to open the *File Transfer* dialog.
2. In the *Send* tab, click the **Add** button. This opens the *Select Files for File Transfer* dialog.
3. In the *Current Workspace* tab, select the top-level *Projects* folder and click **OK** (FIG. 76):

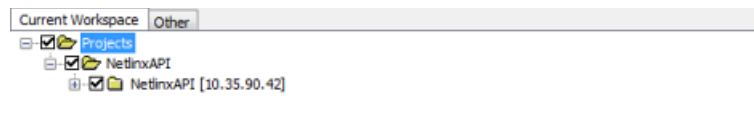


FIG. 76 Select Files for File Transfer dialog - Current Workspace tab

4. Select **OK** to return to the *File Transfer* dialog. The selected files are indicated in the *Send* tab (FIG. 77):

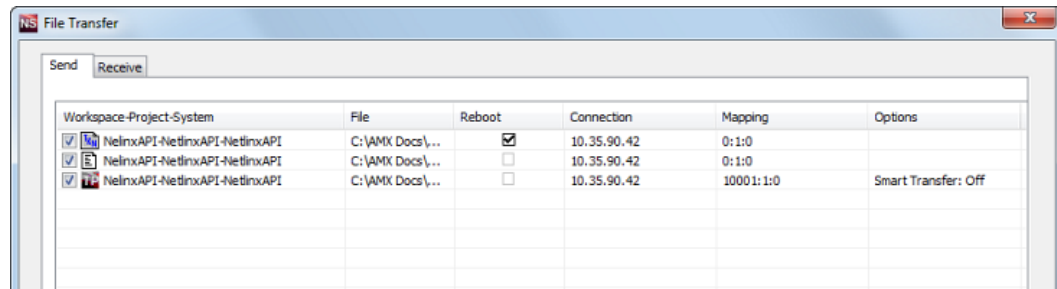


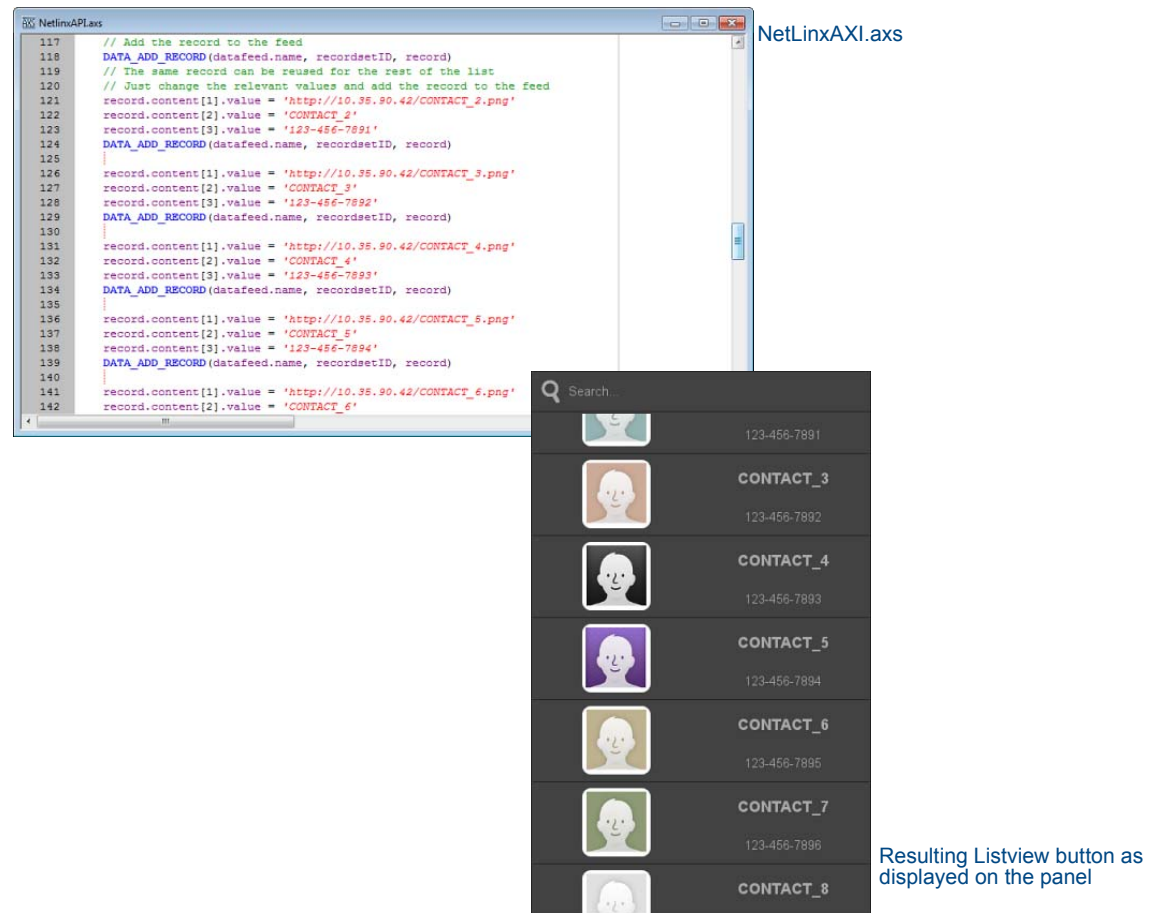
FIG. 77 File Transfer dialog - Send tab

5. In the *File Transfer* dialog, click **Send** to initiate the file transfer.
6. The progress of the transfer is indicated in the Output Bar.

When the transfer is complete, and the NX Master has completed a reboot, the Listview button should appear on the Page it was added to.

## Example 4 (NetLinX Data Source) - Results

FIG. 78 shows an example of a basic Listview button created by following these steps:



**FIG. 78** Example Listview button based on "NetLinXAXI.axs"

Using *NetLinXAXI.axs* file as it's data source:

- It displays the contents of the "record.content[2].value" ID as the *Primary Text* component.
- It displays the contents of the "record.content[3].value" ID as the *Secondary Text* component.
- It displays the contents of the "record.content[1].value" ID as the *Image* component.



While the Listview button shown in this example uses only basic design characteristics, note that Listview buttons support most of the same display options as other button types, including Radiant/Gradient fills, Text Effects, Opacity, etc... Use these options to create eye-catching designs, just like for any other button type.

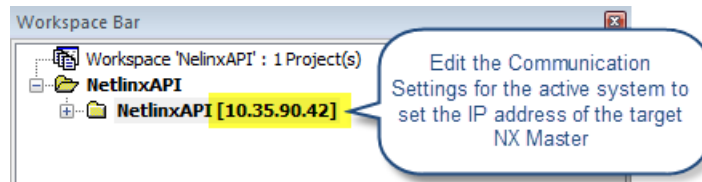
### NetLinxAPI Demo File ("NetLinxAPI.ZIP")

Demo (ZIP) files for the Listview examples presented here are available to download from the UI RESOURCE CENTER at [www.amx.com](http://www.amx.com). The preceding example followed the *NetLinxAPI* demo. The NetLinxAPI demo ZIP file (**NetLinxAPI.zip**) contains the following:

NetLinxAPI ZIP Contents	
File	Description
<b>NetLinxAPI.TP5</b>	TPDesign5 project file that includes a Listview button pre-configured to use the layout properties and data source file shown in the <i>Listview Button/Dynamic Data Example 4: NetLinx Data Source</i> example (see <i>page 56</i> ).
<b>"CONTACT_IMAGES" folder</b>	This folder contains the images used for the Listview button in this example.
<b>NetLinxAPI.apw</b>	NetLinx Studio 4 Workspace file, with the Listview demo custom event defined. This Workspace contains the following files: <ul style="list-style-type: none"> <li>• NetLinxAPI.axs</li> <li>• NetLinxAPI.src</li> </ul>

To use this demo:

1. Download the *NetLinxAPI.ZIP* file and extract it's contents to a known location.
2. In NetLinx Studio 4, open the *NetLinxAPI.apw* workspace file (**File > Open Workspace**).  
This Workspace contains NetLinx source code that is pre-configured with a Custom Event for user selection, as well as a TPDesign5 project that includes a pre-configured Listview button that uses NetLinx data as it's data source.
3. Build the Workspace: Select **Build > Build Active System**.
4. Transfer all files contained in the Workspace to the target NX Master:
  - a. Select **Settings > Active System Communication Settings** to open the *Communication Settings* dialog. Use the options in this dialog to establish a connection to the target NX Master. Note that by default, the workspace is configured to use Serial communication (FIG. 79):



**FIG. 79** NetLinx Studio 4 Select Files for File Transfer dialog (Current Workspace tab) - Projects directory selected

- b. Select **Tools > File Transfer** to open the *File Transfer* dialog (*Send* tab). Remove any files (from previous transfer operations) that may be in the list.
- c. Click **Add** to open the *Select Files for File Transfer* dialog (*Current Workspace* tab).
- d. Click the top-level *Projects* directory to auto-select all files in the Workspace.
- e. Verify that the IP address indicated here indicates the correct NX Master, and click **OK** to save changes and return to the *File Transfer* dialog.
- f. In the *File Transfer* dialog, click **Send** to transfer the Workspace files to the target NX Master.

## Listview (Data Access) Send Commands

The *Data Access Send Commands* described in the following table represent a new set of Button (^) Send Commands that support the use of dynamic data for Listview buttons in NetLinx code. Note that the *variable text address range* (<vt addr range>) indicated in the syntax examples represents the address of the Listview button, and works the same as it does for all other (^) Button Send Commands.

Many Listview Send Commands take a boolean parameter. Any of the following values can be used:

Will resolve to true	Will resolve to false
true	false
TRUE	FALSE
on	off
ON	OFF
1	0
	(empty)

### Terminology

The NetLinx Data Access Send Commands use the following terminology:

NetLinx Data Access Send Commands - Terminology	
Name	Description
<b>DataFeed</b>	A DataFeed is a descriptor with a unique name used to publish data records. A DataFeed can be created by a NetLinx program and then published to the NetLinx web server for external consumption by devices like the G5 touch panel for use with Listview buttons. DataFeeds can also be sourced from a server running the AMX XPort software.
<b>DataRecord</b>	A DataRecord represents a container of data fields and the index/ordinal position of the row in the recordset. A DataRecord may contain metadata and/or content fields.
<b>DataField</b>	A DataField represents the value that stores the actual data elements. All raw data in the NetLinx data access APIs are stored and managed as values and (one or more) attributes.

Listview Commands	
<b>^LVC</b>	<p>Listview Cache Configure - This command configures the image cache used by the Listview.</p> <p>Syntax:</p> <pre>" ^LVC-&lt;configuration_option=configuration_value&gt; "</pre> <p>Variables:</p> <ul style="list-style-type: none"> <li>a comma separated list of one or more configuration parameters followed by an equal sign and the configuration setting.</li> </ul> <p>Configuration Options:</p> <ul style="list-style-type: none"> <li><b>clear</b>- Clear the current memory and disk cache used for Listview image loading.</li> <li><b>mem_size</b> - The size of the memory cache, either as a percentage of the available application memory or as total size. Percentages are specified as floating point. Percentage values are 2% (0.02) to 20% (0.20) and totals are 16 to 256 MB. The default is 10%.(0.10)</li> <li><b>disk_size</b> - The size of the disk cache. Valid values are 16 to 500 MB The default is 200.</li> </ul>

Listview Commands (Cont.)	
<b>^LVD</b>	<p>Listview Data Source - This command sets the data source to drive the Listview entries. Note that this command only configures the data source it does not actually cause the data to be fetched. The ^LVR refresh command (page 68) must be issued to load the data.</p> <p>Syntax:</p> <pre>''^LVD-&lt;vt addr range&gt;,&lt;URL to data source or Dynamic Data Resource name&gt;,&lt;configuration_option=configuration_value&gt;' "</pre> <p>Variables:</p> <ul style="list-style-type: none"> <li>variable text address range = 1 - 4000.</li> <li>URL to the data source/Dynamic Data Resource name (required). If the suffix of the URL is <b>.csv</b> or <b>.CSV</b>, then the URL will be assumed to point to a CSV file. Otherwise the type is assumed to be the XPort amxstandard.xml format. A file on the panel's local file system can be specified using the <b>"file://"</b> option. <b>Note: "ftp://" is not a supported option.</b></li> <li>a optional comma-separated list of one or more configuration parameters followed by an equal sign and the configuration setting.</li> </ul> <p>Configuration Options:</p> <ul style="list-style-type: none"> <li><b>user</b> - The user name to use for authenticating to the web server when retrieving the feed data source file. If specified when URL is a Dynamic Data Resource, this value will override the username inside the Dynamic Data Resource.</li> <li><b>pass</b> - The password to use for authenticating to the web server when retrieving the feed data source file. If specified when URL is a Dynamic Data Resource, this value will override the password inside the Dynamic Data Resource.</li> <li><b>csv</b> - a boolean indicating whether or not to parse the data source as a CSV file. If not present, defaults to false.</li> <li><b>has_headers</b> - a boolean indicating that the first line of the CSV file has column headers which will be used to name the content fields for each data record. If true it automatically implies that csv is also true. If this option is not present then the default for a CSV file is false. In the absence of headers, the content fields will be named using the following convention: <i>column1</i>, <i>column2</i>, <i>column3</i>... (CSV files only, since XML always has field names specified within the file).</li> </ul> <p>Example:</p> <pre>SEND_COMMAND Panel, ''^LVD-42,http://192.168.220.231/public/ lv42data.csv,has_headers=1' "</pre> <p>Configures the Listview button to use the CSV file at the URL as its data source. The first line of the CSV file should be parsed as field names and not as Listview entry record data.</p>
<b>^LVF</b>	<p>Listview Filter - This command can be used to programmatically change the filter contents of the Listview widget. When the filter contents is changed, the filter will be applied to the current Listview data which can change the number of items displayed based on those that meet the filter sequence. The filter changes immediately, and the filter can be set or cleared with this command.</p> <p>Syntax:</p> <pre>''^LVF-&lt;vt addr range&gt;,&lt;filter character sequence&gt;' "</pre> <p>Variables:</p> <ul style="list-style-type: none"> <li>variable text address range = 1 - 4000.</li> <li>filter character sequence. All characters including whitespace characters will be applied to the filter.</li> </ul> <p>Example:</p> <pre>SEND_COMMAND Panel, ''^LVF-42,amx' "</pre> <p>Sets the filter sequence to amx. Only items in the data set that contain the sequence amx will be displayed.</p> <pre>SEND_COMMAND Panel, ''^LVF-42,' "</pre> <p>Clears the filter sequence. All items in the data set can be viewed in the Listview.</p>

Listview Commands (Cont.)																																									
<b>^LVL</b>	<p>Listview Layout - This command sets the layout configuration to configure the visual representation of the Listview entries.</p> <p>Syntax:</p> <pre>''^LVL-&lt;vt addr range&gt;,&lt;layout_option=layout_value&gt;' "</pre> <p>Variables:</p> <ul style="list-style-type: none"> <li>Variable text address range = 1 - 4000.</li> <li>A comma separated list of one or more layout configuration parameters followed by an equal sign and the configuration setting.</li> </ul> <p>Layout Options:</p> <ul style="list-style-type: none"> <li><b>columns</b> - Number of columns parameter. An integer that represents the number of columns to display. The number must be at least 1 and a value that exceeds the minimum cell width will truncate to the maximum.</li> </ul> <p><b>Note:</b> Optional valid tags for the columns parameter are <b>nc=</b>, <b>numcol=</b>, and <b>columns=</b>.</p> <ul style="list-style-type: none"> <li><b>comp</b> - Component parameter. An integer that is a value which determines which graphical components are present in the cell. When the component values are bitwise or'd together, it creates the encoding for the cell components that are populated. If a configuration parameter is not in the current command, the last value for the configuration parameter is used.</li> </ul> <p><b>Note:</b> Optional valid tags for the comp parameter are <b>c=</b> and <b>comp=</b>.</p> <table border="1"> <thead> <tr> <th>Component Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The image (i) is used in the cell.</td></tr> <tr> <td>2</td><td>The primary text field (t1) is used in the cell.</td></tr> <tr> <td>4</td><td>The secondary text field (t2) is used in the cell</td></tr> </tbody> </table> <p>Not all variations of component values are valid. To have the secondary text field present, the primary text field must be preset as well.</p> <table border="1"> <thead> <tr> <th>Component Combinations</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Invalid. No component displayed.</td></tr> <tr> <td>1</td><td>The image (i) is the only component displayed.</td></tr> <tr> <td>2</td><td>The primary text field (t1) is the only component displayed.</td></tr> <tr> <td>3</td><td>The image (i) and the primary text field (t1) are displayed.</td></tr> <tr> <td>4</td><td>Secondary text (t2) only. Invalid. Secondary text (t2) cannot be displayed without the primary text (t1).</td></tr> <tr> <td>5</td><td>Secondary text (t2) and image (i). Invalid. Secondary text (t2) cannot be displayed without the primary text (t1).</td></tr> <tr> <td>6</td><td>The primary text (t1) and secondary text (t2) are displayed.</td></tr> <tr> <td>7</td><td>The image (i), primary text (t1), and secondary text (t2) are displayed</td></tr> </tbody> </table> <ul style="list-style-type: none"> <li><b>cellheight</b> - An integer or percentage that sets the height of a cell. The value can be an integer &gt;= the minimum cell height (48), or a percentage of the list height (5% up to 95%). To specify a percentage, append a '%' to the end of the value.</li> </ul> <p><b>Note:</b> Valid tags for the cellheight param are <b>ch=</b> and <b>cellheight=</b>.</p> <ul style="list-style-type: none"> <li><b>layout</b> - An integer that sets the layout configuration of each cell.</li> </ul> <p><b>Note:</b> valid tags for the layout parameter are <b>l=</b> and <b>layout=</b>.</p> <table border="1"> <thead> <tr> <th>Layout Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Horizontal layout with image on the left and text(s) on the right. If multiple texts are selected then the texts are stacked vertically</td></tr> <tr> <td>2</td><td>Horizontal layout with image on the right and text(s) on the left. If multiple texts are selected then the texts are stacked vertically.</td></tr> <tr> <td>3</td><td>Horizontal layout with text1 on the left, image in the center, and text2 on the right. If multiple texts are selected then the texts are stacked vertically.</td></tr> <tr> <td>4</td><td>Vertical layout with the image on the top and text(s) below the image. If multiple texts are selected then text1 is below the image and text2 is below text1.</td></tr> <tr> <td>5</td><td>Vertical layout with the image on the bottom and text(s) above the image. If multiple texts are selected then text1 is on top, text2 is below text1, and the image is below text2.</td></tr> <tr> <td>6</td><td>Vertical layout with text1 on top, the image below text1, and text2 below the image.</td></tr> </tbody> </table>	Component Value	Description	1	The image (i) is used in the cell.	2	The primary text field (t1) is used in the cell.	4	The secondary text field (t2) is used in the cell	Component Combinations	Description	0	Invalid. No component displayed.	1	The image (i) is the only component displayed.	2	The primary text field (t1) is the only component displayed.	3	The image (i) and the primary text field (t1) are displayed.	4	Secondary text (t2) only. Invalid. Secondary text (t2) cannot be displayed without the primary text (t1).	5	Secondary text (t2) and image (i). Invalid. Secondary text (t2) cannot be displayed without the primary text (t1).	6	The primary text (t1) and secondary text (t2) are displayed.	7	The image (i), primary text (t1), and secondary text (t2) are displayed	Layout Value	Description	1	Horizontal layout with image on the left and text(s) on the right. If multiple texts are selected then the texts are stacked vertically	2	Horizontal layout with image on the right and text(s) on the left. If multiple texts are selected then the texts are stacked vertically.	3	Horizontal layout with text1 on the left, image in the center, and text2 on the right. If multiple texts are selected then the texts are stacked vertically.	4	Vertical layout with the image on the top and text(s) below the image. If multiple texts are selected then text1 is below the image and text2 is below text1.	5	Vertical layout with the image on the bottom and text(s) above the image. If multiple texts are selected then text1 is on top, text2 is below text1, and the image is below text2.	6	Vertical layout with text1 on top, the image below text1, and text2 below the image.
Component Value	Description																																								
1	The image (i) is used in the cell.																																								
2	The primary text field (t1) is used in the cell.																																								
4	The secondary text field (t2) is used in the cell																																								
Component Combinations	Description																																								
0	Invalid. No component displayed.																																								
1	The image (i) is the only component displayed.																																								
2	The primary text field (t1) is the only component displayed.																																								
3	The image (i) and the primary text field (t1) are displayed.																																								
4	Secondary text (t2) only. Invalid. Secondary text (t2) cannot be displayed without the primary text (t1).																																								
5	Secondary text (t2) and image (i). Invalid. Secondary text (t2) cannot be displayed without the primary text (t1).																																								
6	The primary text (t1) and secondary text (t2) are displayed.																																								
7	The image (i), primary text (t1), and secondary text (t2) are displayed																																								
Layout Value	Description																																								
1	Horizontal layout with image on the left and text(s) on the right. If multiple texts are selected then the texts are stacked vertically																																								
2	Horizontal layout with image on the right and text(s) on the left. If multiple texts are selected then the texts are stacked vertically.																																								
3	Horizontal layout with text1 on the left, image in the center, and text2 on the right. If multiple texts are selected then the texts are stacked vertically.																																								
4	Vertical layout with the image on the top and text(s) below the image. If multiple texts are selected then text1 is below the image and text2 is below text1.																																								
5	Vertical layout with the image on the bottom and text(s) above the image. If multiple texts are selected then text1 is on top, text2 is below text1, and the image is below text2.																																								
6	Vertical layout with text1 on top, the image below text1, and text2 below the image.																																								

Listview Commands (Cont.)															
^LVF (Cont.)	<ul style="list-style-type: none"><li><b>layout</b> - An integer that sets the layout configuration of each cell. <i>Note: valid tags for the layout parameter are <b>l=</b> and <b>layout=</b>.</i></li></ul>														
	<table><tr><th>Layout Value</th><th>Description</th></tr><tr><td>1</td><td>Horizontal layout with image on the left and text(s) on the right. If multiple texts are selected then the texts are stacked vertically</td></tr><tr><td>2</td><td>Horizontal layout with image on the right and text(s) on the left. If multiple texts are selected then the texts are stacked vertically.</td></tr><tr><td>3</td><td>Horizontal layout with text1 on the left, image in the center, and text2 on the right. If multiple texts are selected then the texts are stacked vertically.</td></tr><tr><td>4</td><td>Vertical layout with the image on the top and text(s) below the image. If multiple texts are selected then text1 is below the image and text2 is below text1.</td></tr><tr><td>5</td><td>Vertical layout with the image on the bottom and text(s) above the image. If multiple texts are selected then text1 is on top, text2 is below text1, and the image is below text2.</td></tr><tr><td>6</td><td>Vertical layout with text1 on top, the image below text1, and text2 below the image.</td></tr></table>	Layout Value	Description	1	Horizontal layout with image on the left and text(s) on the right. If multiple texts are selected then the texts are stacked vertically	2	Horizontal layout with image on the right and text(s) on the left. If multiple texts are selected then the texts are stacked vertically.	3	Horizontal layout with text1 on the left, image in the center, and text2 on the right. If multiple texts are selected then the texts are stacked vertically.	4	Vertical layout with the image on the top and text(s) below the image. If multiple texts are selected then text1 is below the image and text2 is below text1.	5	Vertical layout with the image on the bottom and text(s) above the image. If multiple texts are selected then text1 is on top, text2 is below text1, and the image is below text2.	6	Vertical layout with text1 on top, the image below text1, and text2 below the image.
	Layout Value	Description													
	1	Horizontal layout with image on the left and text(s) on the right. If multiple texts are selected then the texts are stacked vertically													
	2	Horizontal layout with image on the right and text(s) on the left. If multiple texts are selected then the texts are stacked vertically.													
	3	Horizontal layout with text1 on the left, image in the center, and text2 on the right. If multiple texts are selected then the texts are stacked vertically.													
	4	Vertical layout with the image on the top and text(s) below the image. If multiple texts are selected then text1 is below the image and text2 is below text1.													
	5	Vertical layout with the image on the bottom and text(s) above the image. If multiple texts are selected then text1 is on top, text2 is below text1, and the image is below text2.													
	6	Vertical layout with text1 on top, the image below text1, and text2 below the image.													
	<ul style="list-style-type: none"><li><b>p1</b> - layout percentage 1. Sets the boundaries between cell components in different layouts. An integer between 10 and 90 that sets the boundary between components as a percentage of the cell dimension. The percentage can be specified as a number between 5-95 with an optional percentage sign '%' at the end.</li><li><b>p2</b> - layout percentage 2. Sets the boundaries between cell components in different layouts. An integer between 10 and 90 that sets the boundary between components as a percentage of the cell dimension. The percentage can be specified as a number between 5-95 with an optional percentage sign '%' at the end.</li><li><b>filter</b> - Enable or disable the search filter on the Listview. To enable set to 'true', 'on', or '1'. To disable set to 'false', 'off', or '0'. <i>Note: Valid tags for the filter parameter are <b>f=</b> and <b>filter=</b>.</i></li><li><b>filterheight</b> - An integer or percentage that sets the height of the filter in the Listview. The value can be an integer &gt;= the minimum filter height (24), or a percentage of the list height (5% to 25%). To specify a percentage, append a '%' to the end of the value. <i>Note: Valid tags for the filterheight param is <b>fh=</b> and <b>filterheight=</b>.</i></li><li><b>alphascroll</b> - Enable or disable the alpha scroll on the Listview. To enable set to 'true', 'on', or '1'. To disable set to 'false', 'off', or '0'. (NOTE: Valid tags for the alphascroll parameter are <b>as=</b> and <b>alphascroll=</b>).</li></ul>														
Examples:															
<pre>SEND_COMMAND Panel,"'^LVL-42, layout=1, comp=7, columns=1, cellheight=120, p1=40%, p2=66%' "</pre> <p>Sets the Listview configuration display an image and 2 text fields (comp=7), in a layout 1 configuration (layout=1 horizontal layout of the image on left and text1 and text2 to the right of the image). There is 1 column (columns=1) and the cell is 120 pixels high (h=120). The image width will be 40% of the cell width (p1=40%) with text1 and text2 having a width of 60% of the cell width. The height of text1 will be 66% of the cell height (p2=66%) with text2 height of 34% of the cell height.</p>															
<pre>SEND_COMMAND Panel,"'^LVL-42,l=4, c=3, ch=150, nc=4, p1=70%' "</pre> <p>Sets the Listview configuration display an image and 1 text fields (c=4), in a layout 4 configuration (l=4 vertical layout of the image on top and text1 below the image). There are 4 columns (nc=4) and the cell is 150 pixels high (ch=150). The image height will be 70% of the cell height (p1=70) with text1 having a height of 30% of the cell height.</p>															
<pre>SEND_COMMAND Panel,"'^LVL-42,layout=3, comp=6, ch=100, numcol=1, p1=50%' "</pre> <p>Sets the Listview configuration display 2 text fields (comp=6), in a layout 3 configuration (layout=2 horizontal layout of text1 on the left and text2 on the right). There is 1 column (numcol=1) and the cell is 100 pixels high (ch=100). The text1 width will be 50% of the cell width (p1=50) with text2 having a width of 50% of the cell width.</p>															
<pre>SEND_COMMAND Panel,"'^LVL-42,filter=1, fh=10%, as=false%' "</pre> <p>Sets the Listview search filter enabled (filter=1), the search filter textview height to 10% of the Listview height (fh=10%), and disables the alphascroller on the Listview.</p>															



Listview Commands (Cont.)	
<b>^LVM</b>	<p>Listview Map Fields - This command maps the fields from the data source to the display elements of a Listview entry. Each list entry corresponds to a record if the data came from the NetLinX data access API or XPort. If the data source is a csv file, then each list entry corresponds to a row in the file. A list entry can have up to two lines of text and a URL that points to an image. Each display element for a list entry has to be mapped to a field in the record. If no mapping is specified, then a default mapping is used which is simply to map the fields in order based on the screen layout of the list entry. So, if the list type was an image and two lines of text, the first content field in the record would be interpreted as the URL to the image, the next field would be the first line of text and the next field would be the second line of text. To override this default behavior, the ^LVM command should be used to specify the correct mapping.</p> <p>Syntax:</p> <pre>''^LVM-&lt;vt addr range&gt;,&lt;display_element=field_expression  &lt;display_element=field_expression&gt; ...''</pre> <p>Variables:</p> <ul style="list-style-type: none"> <li>• variable text address range = 1 - 4000.</li> <li>• a pipe character " " separated list of mapping expressions. A pipe is used because typical field expressions may use more common characters such as the comma or semicolon.</li> </ul> <p>Display Elements:</p> <ul style="list-style-type: none"> <li>• t1 - the first text element</li> <li>• t2 - the second text element</li> <li>• i1 - the first image</li> <li>• future display types may support more text and image elements which will follow the same convention: t3... i2...</li> </ul> <p>Field Expressions:</p> <p>An expression that can be used to map field values to display elements. Any time a field name is used, it follows the form <b>\$(field_name)</b>. Other text characters can be used to construct a more complex string using multiple fields.</p> <p>Examples:</p> <pre>SEND_COMMAND Panel, ''^LVM-42,i1=\$(image)''</pre> <p>Configures the Listview widget to map an image field to the image display element. In this example, the Listview type is assumed to be a single image only.</p> <pre>SEND_COMMAND Panel, ''^LVM-42,i1=\$(image) t1=\$(lname), \$(fname) t2=\$(number)''</pre> <p>The Listview widget is the type that has an image and two lines of text. The top line will consolidate two different fields in the form of last name, first name. The second line of text will be the phone number.</p> <pre>SEND_COMMAND Panel, ''^LVM-42,t1=\$(column2), \$(column1) t2=\$(column3) i1=\$(column4)''</pre> <p>This is the same example as the one above it but the source of the data was a csv file that didn't have any headers. The csv columns were laid out as first name, last name, number, URL to image.</p>
<b>^LVN</b>	<p>Listview Navigate - This command can be used to move the Listview widget. Navigation commands will be range checked. The command will attempt to position the specified list entry on the top line of the Listview widget. When navigating at the end of the list, however, the widget will position the last item in the list on the bottom line and will not leave blank lines at the bottom. The only exception to this case will be when the Listview has fewer entries than the number of displayable entries. If the optional select boolean is present, and the navigation command used support the select option, the item at the destination will be selected and a item selected custom event will be initiated.</p> <p>Syntax:</p> <pre>''^LVN-&lt;vt addr range&gt;,&lt;navigation_command&gt;,[optional boolean_select_param]''</pre> <p>Variables:</p> <ul style="list-style-type: none"> <li>• variable text address range = 1 - 4000.</li> <li>• navigation command.</li> <li>• optional select boolean</li> </ul>

Listview Commands (Cont.)	
<b>^LVN</b> (Cont.)	<p>Navigation Commands:</p> <ul style="list-style-type: none"> <li>• t or T - move to the top of the list (supports an optional select boolean).</li> <li>• b or B - move to the bottom of the list (supports an optional select boolean).</li> <li>• d or D - page down (DOES NOT support the optional select boolean. A select boolean will be ignored if present).</li> <li>• n - move to a specific list entry number at position n. n is a zero based index. (supports an optional select boolean). (Note: If n is &lt; 0 and select is true then the current selected item is deselected.)</li> <li>• u or U - page up (DOES NOT support the optional select boolean. A select boolean will be ignored if present).</li> </ul> <p>Examples:</p> <pre>SEND_COMMAND Panel, "'^LVN-42,B'"     Move to the bottom of the list. SEND_COMMAND Panel, "'^LVN-42,d'"     Move the list down a page. SEND_COMMAND Panel, "'^LVN-42,3,1'"     Move the list to position 3 in the list and select the item at position 3.</pre>
<b>^LVR</b>	<p>Listview Refresh Data - This command has two different functions. If it is sent without any parameters, it causes the Listview widget to load data from its configured data source. If optional parameters are included with the command, then the automatic data refresh options are configured.</p> <p>The typical behavior for auto refresh is that the last modified time of the data source is tracked. At the refresh interval, the last modified time of the data source is compared against the stored value.</p> <p>If the data is newer, then it is reloaded and the Listview widget is refreshed with the updated data. If the data is unchanged, then it is not reloaded. The default for auto refresh is off.</p> <p>Syntax:</p> <pre>''^LVR-&lt;vt addr range&gt;,[optional refresh_interval],[optional force_reload]''</pre> <p>Variables:</p> <ul style="list-style-type: none"> <li>• variable text address range = 1 - 4000.</li> <li>• refresh_interval - the optional interval (in seconds) at which to check for newer data. 0 (the default) means auto refresh is off. Minimum is 5 seconds. If not specified, the current refresh interval is retained.</li> <li>• force_reload - the optional parameter to force the Listview to ignore and data file timestamps and to force a clear on image caches for refreshed Listview images. Not specified or 0 will not force a reload, 1 will force a reload of data file and images associated with data file. (Note: This can cause the images in a Listview to flicker upon the reload. This is the expected behavior due to the images being reloaded from the server.)</li> </ul> <p>Example:</p> <pre>SEND_COMMAND Panel, "'^LVR-42'"     Commands the Listview widget to load the data from the data source and populate the Listview display widget. SEND_COMMAND Panel, "'^LVR-42,15'"     Commands the Listview widget to check for an updated data source every 15 seconds. SEND_COMMAND Panel, "'^LVR-42,600,1'"     Commands the Listview widget to check for an updated data source every hour, and to force a reload of the data and the images.</pre>

Listview Commands (Cont.)	
<b>^LVS</b>	<p>Listview Sort Data - This command sets the columns that are used for sorting of lists, as well as the type of sorting that is done. The multiple columns are allowed in the sort procedure. The order of the columns in the command determine the order of the sorting. The first column is the primary sorting data, the second would be used for sorting with rows of data that are equal in the primary columns, and so on for however many columns are used for sorting. If no columns are listed in the command, then the current sorting columns are used if they have been previously defined.</p> <p>The type of sort is an optional part of the command and follows the sort columns. Initially, there are four different sort types available.</p> <ul style="list-style-type: none"> <li>• None (n) - No sorting is performed.</li> <li>• Ascending (a) - Ascending sort using localized character weighting.</li> <li>• Descending (d) - Descending sort using localized character weighting.</li> <li>• Override (*) - Override sort syntax portion of command determines sorting.</li> </ul> <p>The override sort syntax allows for complex SQLite ORDER BY syntax for sorting. When override is selected, the sort columns that were set in the command or previously are ignored and the entire sorting statement must be in the override sort syntax. The words ORDER BY should not be in the syntax. They are inserted by the firmware.</p> <p>Syntax:</p> <pre>''^LVS-&lt;vt addr range&gt;,&lt;primary sort column name, secondary sort column name,..., final sort column name&gt;,[optional sort type],[optional override sort syntax]''</pre> <p>Variables:</p> <ul style="list-style-type: none"> <li>• variable text address range = 1 - 4000.</li> <li>• Sort columns - comma separated list of sort columns in the order of sort priority. Sort columns can be specified using the \${column name} syntax that is used in the ^LVM command. Columns can be Content Fields or Metadata Fields in the master Datafeed XML file generated by the master. Metadata fields are prepended with "meta" in front of the "ID" attribute of the field.</li> <li>• Sort Type - A character indicating the sorting algorithm to use. <ul style="list-style-type: none"> <li>'a' - ascending</li> <li>'d' - descending</li> <li>'*' - override. Sort command syntax must follow in the next part of the command.</li> <li>'n' - none (default). Any character that is not a,d, or * will set sort to none.</li> </ul> </li> <li>• Override sort syntax - A SQLite ORDER BY statement to use as the sort.</li> </ul> <p>Examples:</p> <pre>SEND_COMMAND Panel, ''^LVS-42, \${artist name},\${title};a ''</pre> <p>Commands the Listview widget to sort the data source by the artist name and then title in an ascending order. Equates to "artistname, title COLLATE LOCALIZED ASC" override syntax.</p> <pre>SEND_COMMAND Panel, ''^LVS-42, \${artist name},\${title};d ''</pre> <p>Commands the Listview widget to sort the data source by the artist name and then title in an descending order. Equates to "artistname COLLATE LOCALIZED DESC, title COLLATE LOCALIZED DESC" override syntax.</p> <pre>SEND_COMMAND Panel, ''^LVS-42,;n''</pre> <p>Commands the Listview widget to not sort the current data.</p> <pre>SEND_COMMAND Panel, ''^LVS-150,\${user name},\${text};*;meta\${Record timestamp} ASC''</pre> <p>Commands the panel to sort by the meta data field Record timestamp in ASCENDING order. The username and test fields are ignored.</p> <pre>SEND_COMMAND Panel, ''^LVS-150,;*;meta\${Record timestamp} ASC''</pre> <p>Commands the panel to sort by the meta data field "Record timestamp" in ASCENDING order. The username and test columns are ignored.</p> <pre>SEND_COMMAND Panel, ''^LVS-150,;*;LENGTH(\${description}),\${description} ASC''</pre> <p>Command the panel to sort by the number of characters in the description field, and then by the contents of the description field in ASCENDING order</p>



# Listview Button Properties

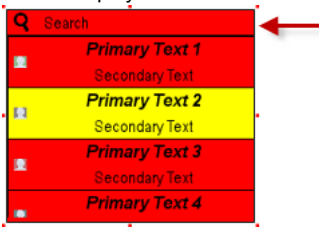
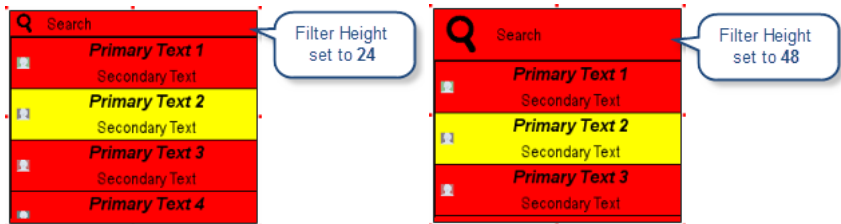
## Overview

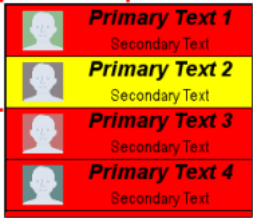
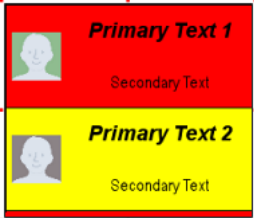
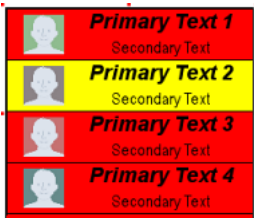

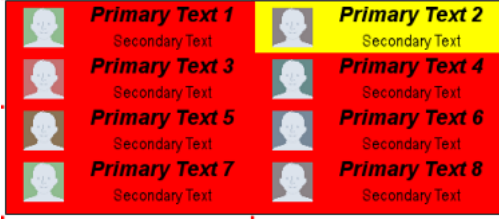
The Properties presented in the Properties window will depend on the current selection in the active Design View window (Page, Popup Page, Sub-Page, Application window or Button). The properties described in this section are specific to Listview Buttons and are provided in this document as reference for the Listview Buttons and Dynamic Data demos.



For descriptions of all of the properties available in TPDesign5, refer to the TPDesign5 online help or Instruction Manual.

## General Properties:

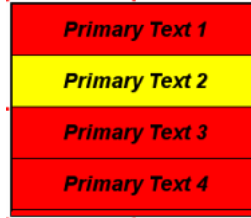
Listview Button-Specific General Properties	
<b>Alphabet Scrollbar</b>	<p>Use this property to enable/disable the alphabet scrollbar feature for Listview buttons. By default, this property is set to no (disabled).</p> <p>To enable this feature, select yes from the drop-down menu. If enabled, an alphabetical index will be rendered on the Listview button. This allows the end user to quickly scroll through a large numbers of list items, with an indication of where the user is in the current view, relative to the alphabetic order of the list items.</p> <p>Note that the scrollbar is not visible in TPDesign5, it is only rendered on the panel (when enabled).</p>
<b>Dynamic Data Source</b>	<p>Use this property to specify the data source to use as the source for content that will be displayed on the selected Listview button:</p> <p>Click the browse button on the Dynamic Data Source property to open the <i>Select Resource</i> dialog. Use the <i>Select Resource</i> dialog to specify which components (Primary Text, Secondary Text and/or Image) will be displayed.</p>
<b>Filter Enabled</b>	<p>Use this property to enable/disable the filter (Search) feature on the selected Listview button. By default, this property is set to no (disabled).</p> <p>To enable this feature, select <b>yes</b> from the drop-down menu. If enabled, a search window will be rendered at the top of the Listview button, with a height specified by the Filter Height property (see below). The remaining area of the Listview button will be available for the display of list items:</p> 
<b>Filter Height</b>	<p>Use this property to specify the height of the filter entry box for a Listview button (in pixels). The minimum allowed value (and the default setting) is 24 pixels.</p> 

Listview Button-Specific General Properties (Cont.)	
<b>Item Height</b>	<p>Use this property to specify the item height for the selected Listview button (in pixels). Note that all list items are drawn to the height specified here, regardless of the overall size of the Listview button itself. That is, adjusting the size of the button does not affect the size of the list items, only the number of list items that can be displayed within the button at a time. The end user will typically need to scroll vertically through the list to see all list items.</p> <div style="display: flex; justify-content: space-around; align-items: flex-end;"> <div style="text-align: center;">  <p>Example: Item Height set to 48</p> </div> <div style="text-align: center;">  <p>Example: Item Height set to 96</p> </div> </div> <p>The minimum allowed value (and the default setting) is 48 pixels.</p>
<b>Listview Columns</b>	<p>Use this property to specify the number of columns to display in the selected Listview button. By default, this value is set to 1. This property provides the ability to present a "grid view" on the Listview button, if desired.</p> <p>The number of columns allowed depends on the size of the Listview button. If the number of columns exceeds the display area of the selected Listview button, the program displays an error message indicating that either the number of columns must be reduced, or the width of the button must be increased to accommodate the desired number of columns.</p> <p>The following example shows a Listview button with one column, two columns, and three columns. Note that the example for three columns displays the Image component only, as set via the <i>Listview Components</i> (General) property.</p> <div style="display: flex; justify-content: space-around; align-items: flex-end;"> <div style="text-align: center;">  <p>Listview Columns set to 1</p> </div> <div style="text-align: center;">  <p>Listview Columns set to 3 (note that this example uses only the Image component)</p> </div> </div> <div style="display: flex; justify-content: space-around; align-items: flex-end; margin-top: 10px;"> <div style="text-align: center;">  <p>Listview Columns set to 2 (button has been resized to accommodate 2 columns)</p> </div> </div>
<b>Listview Components</b>	<p>Use this property to specify the combination of the three supported Components (Primary Text, Secondary Text and/or Image) that will be displayed for list item content on the selected Listview button.</p> <p>Click the browse button on the <i>Listview Components</i> (General) property to open the <i>Edit Listview Components</i> dialog. Use this dialog to specify which components (<i>Primary Text</i>, <i>Secondary Text</i> and <i>Image</i>) will be displayed on the selected Listview button.</p> <ul style="list-style-type: none"> <li>If only <b>Primary Text</b> is selected in the <i>Edit Listview Components</i> dialog (the default setting for new Listview buttons), each list item is represented with a single line of text using center-middle justification and the font face and size specified by the <i>Text Color</i>, <i>Font</i> and <i>Font Size</i> (State) properties (as well as <i>Text Effect</i> and <i>Text Effect Color</i> if desired).</li> </ul>

### Listview Button-Specific General Properties (Cont.)

#### Listview Components (Cont.)

The *Listview Components* (General) property will indicate **single-line text**:



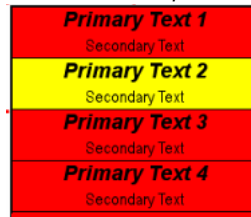
Listview Components single-line text ...

- If **Primary Text** and **Secondary Text** are selected in the *Edit Listview Components* dialog, each list item is represented with two lines of text. The two lines of text are stacked vertically, with each line centered horizontally. The font face and size are specified by the *Secondary Font* and *Secondary Font Size (State)* properties. The text is rendered within a two-pixel margin of the button boundary.

Note that the *Secondary Text* option is only enabled if *Primary Text* is selected.

*Secondary Text* uses the same Text Color settings as the *Primary Text*.

The *Listview Components* property will indicate **two-line text**:



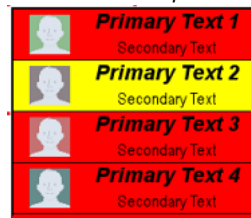
Listview Components two-line text ...

- If **Primary Text**, **Secondary Text** and **Image** are selected in the *Edit Listview Components* dialog, each list item is represented with two lines of text and an image on the left side.

The image is left-justified within a six-pixel margin of the top, bottom, and left item boundaries, and is scaled-to-fit within a square region.

The two lines of text are stacked vertically and centered horizontally in the remaining item region. The top line (*Primary Text*) is rendered using the font face and size specified by the *Font* and *Font Size (State)* properties. The bottom line (*Secondary Text*) is rendered using the font face and size specified by the *Secondary Font* and *Secondary Font Size (State)* properties. The text is rendered within a two-pixel margin of the button boundary.

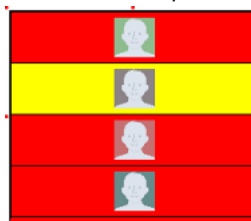
The *Listview Components* Property will indicate **two-line text w/ Image**:



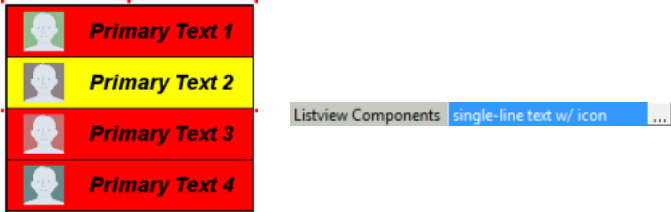
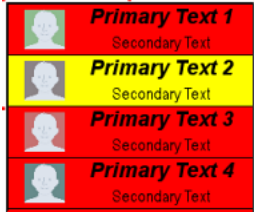
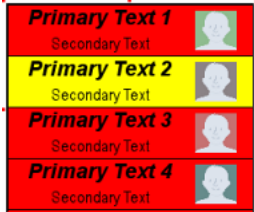
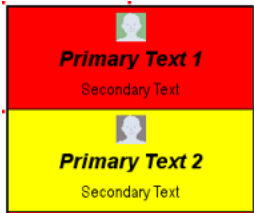
Listview Components two-line text w/ icon ...

- If only **Image** is selected in the *Edit Listview Components* dialog, each list item is represented with a single image centered horizontally within the item region, within a six-pixel margin of the item region.

The *Listview Components* Property will indicate **image only**:



Listview Components icon only ...

Listview Button-Specific General Properties (Cont.)	
<b>Listview Components</b> (Cont.)	<ul style="list-style-type: none"> <li>If <b>Primary Text</b> and <b>Image</b> are selected in the <i>Edit Listview Components</i> dialog, each list item is represented with a single line of text and an image on the left side. The image is left-justified within a six-pixel margin of the top, bottom, and left item boundaries, and is scaled-to-fit within a square region. The text is center-middle justified in the remaining portion of the item region within a two-pixel margin, using the font and font size specified by the <i>Font</i> and <i>Font Size</i> (States) properties. The <i>Listview Components</i> Property will indicate <b>single-line text w/ Image</b>:             <div data-bbox="667 464 1333 674">  </div> </li> </ul>
<b>Listview Item Layout</b>	<p>Use this property to specify the layout of the components (Primary Text, Secondary Text and Image) specified to display on the list items in the selected Listview button. Listview components are selected via the <i>List View Components</i> (General) property. Click in the Listview Item Layout field to select from a drop-down of layout options for list items:</p> <ul style="list-style-type: none"> <li><i>horizontal - image left</i> (default setting): The image (if displayed) will appear to the left of the Primary (and Secondary) Text:             <div data-bbox="667 894 919 1104">  </div> </li> <li><i>horizontal - image right</i>: The image will appear to the right of the text:             <div data-bbox="667 1146 919 1356">  </div> </li> <li><i>vertical - image top</i>: The image will appear centered above the text:             <div data-bbox="667 1398 919 1608">  </div> </li> </ul> <p><b>Note:</b> Once the Listview Item Layout has been selected, the placement of the layout components can be adjusted via the <i>Primary Partition (%)</i> and <i>Secondary Partition (%)</i> properties. In these examples, adjustments have been made to both the partition (%) properties and in the case of the vertical layout example, the <i>Item Height (General)</i> property was adjusted to increase the height to allow the display of all three components.</p>



### Listview Button-Specific General Properties (Cont.)

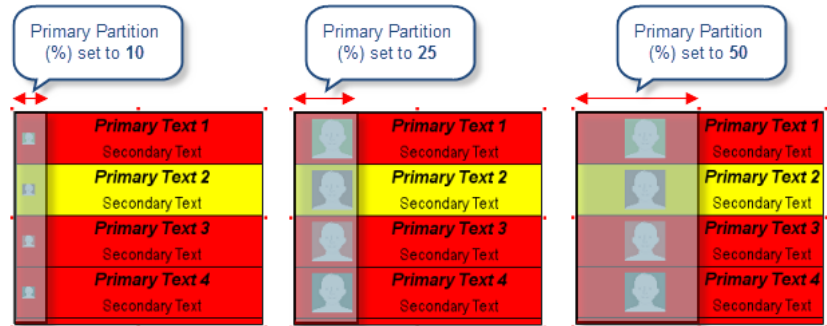
#### Primary Partition (%)

Use this property to specify the relative size of the Primary Partition on the list items displayed on the selected Listview button. The allowed range is 5-95%.

The portion of the list item that is controlled by the Primary Partition (%) property depends on the Listview Components selected, as well as the Listview Item Layout selected. Note that for all layout options, if Image is not an included component, then Primary Partition (%) is ignored. The following examples show a Listview button with all Listview Components (Primary Text, Secondary Text and Image) selected.

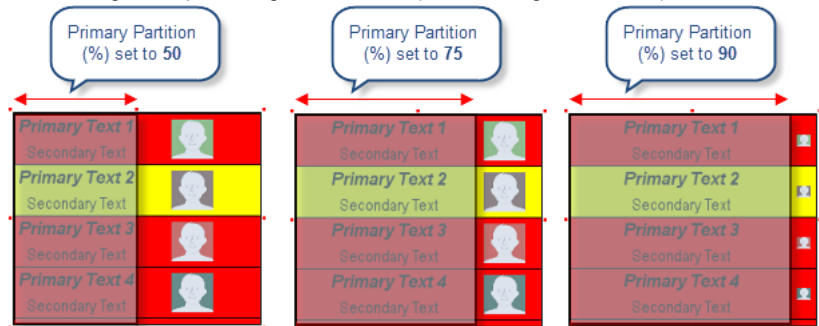
#### With "horizontal - image left" layout selected:

Primary Partition (%) represents the area used by the Image component:



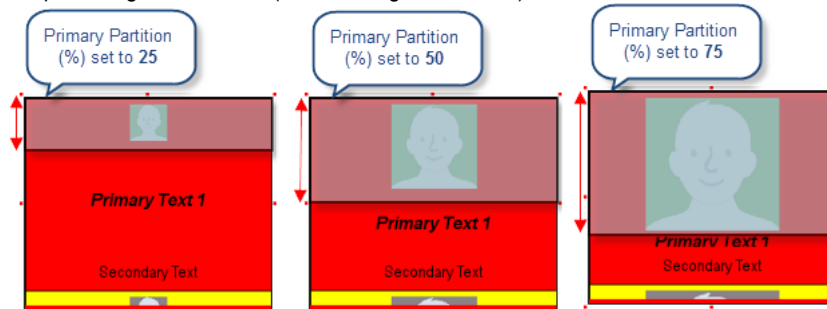
#### With "horizontal - image right" layout selected:

Primary Partition (%) represents the area used by the Primary Text component. In this case, Primary Partition (%) sets the position of the separation between the Primary Text and the Image as a percentage of cell width (allowed range = 5%-95%):



#### With "vertical - image top" selected:

Primary Partition (%) represents the area used by the Image. In this case, Primary Partition (%) sets the position of the separation between the Image and the Primary Text as a percentage of cell width (allowed range = 5%-95%):



### Listview Button-Specific General Properties (Cont.)

#### Secondary Partition (%)

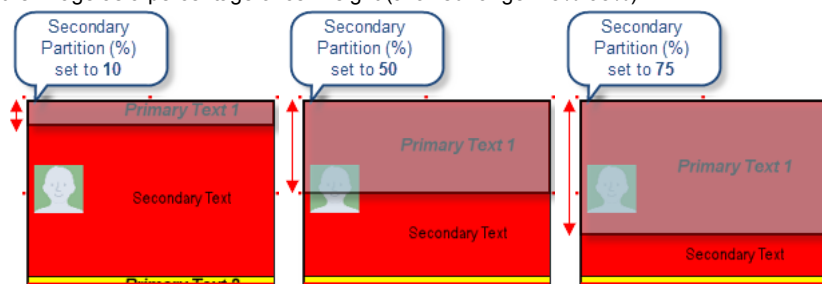
Use this property to specify the relative size of the Secondary Partition on the list items displayed on the selected Listview button. The allowed range is 5-95%.

The portion of the list item that is controlled by the Secondary Partition (%) property will depend on the Listview Components selected as well as the Listview Item Layout selected.

The following examples show a Listview button with all Listview Components (Primary Text, Secondary Text and Image) selected.

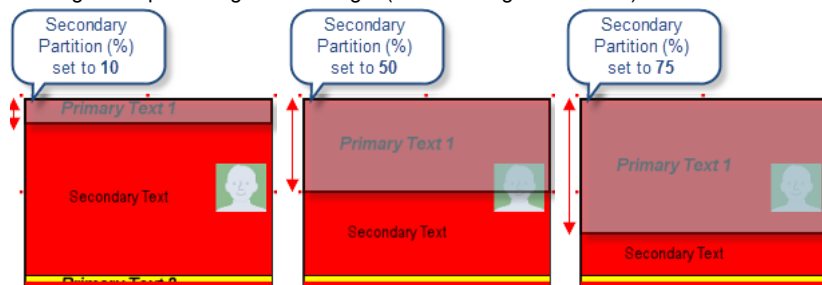
#### With "horizontal - image left" layout selected:

Secondary Partition (%) sets the position of the separation between the Primary Text and the Image as a percentage of cell height (allowed range = 5%-95%):



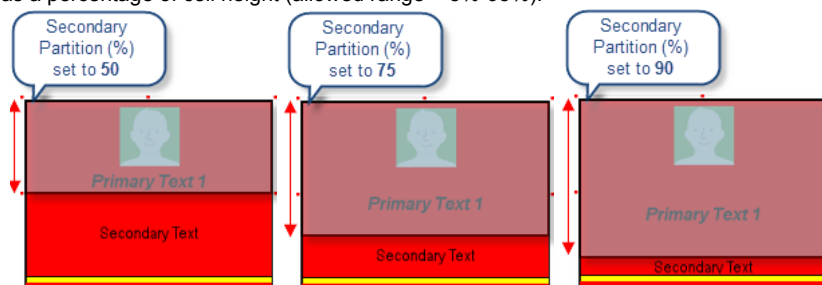
#### With "horizontal - image right" layout selected:

Secondary Partition (%) sets the position of the separation between the Primary Text and the Image as a percentage of cell height (allowed range = 5%-95%):

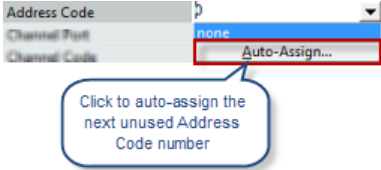



#### With "vertical - image top" selected:

Secondary Partition (%) represents the area used by the Image. In this case, Secondary Partition (%) sets the position of the separation between the Image and the Primary Text as a percentage of cell height (allowed range = 5%-95%):



## Programming Properties.

Listview Button-Specific Programming Properties	
<b>Address Port</b>	<p>Select or enter the port to which the selected element's Address Code will be associated. The options are "1" (the default setting) and "0-setup port":</p> <ul style="list-style-type: none"> <li>If 1 is selected as the Address Port, then the options for the Address Code property are None and Auto-Assign.</li> <li>If 0-Setup Port is selected as the Address Port, then the options for Address Code are Advanced Codes or Basic Codes. By default, the Basic Address Codes are displayed. See Address Codes (Basic and Advanced).</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>The combination of Address Port and Address Code must be unique.</li> <li>Address Port and Address Code assignments for Sub-Pages and Sub-Page View Buttons are provided only for use in SEND-COMMANDS (not to trigger actions).</li> </ul>
<b>Address Code</b>	<p>Select or enter the address code sent to the master on the specified Address Port. The options available to the Address Code property depend on the Address Port selection:</p> <ul style="list-style-type: none"> <li>If 1 is selected as the Address Port, then the options for Address Code are None and Auto-Assign.           <p>Select <b>None</b> to leave the Address Code unspecified.</p> <p>Select <b>Auto-Assign</b> to automatically assign the next available Address Code to the selected TPD5 element.</p>  </li> <li>If <b>0-Setup Port</b> is selected as the Address Port, then the options for Address Code are <i>Advanced Codes</i> or <i>Basic Codes</i>. By default, the Basic Address Codes are displayed:           <p>Address Port set to 0 - setup port</p>  <p>Click on <i>Date Display</i> to select from a list of date display formats.</p> <p>Click on <i>Time Display</i> to select from a list of time display formats.</p> <p>Click <b>Advanced Codes</b> to view the Advanced Channel Code options:</p> </li> </ul>

Listview Button-Specific Programming Properties (Cont.)	
<b>Address Code (Cont.)</b>	<p>Click on <i>None</i> to leave the Address Code unspecified.</p> <p>Click on <i>Panel Setup</i> to select <b>Connection Status</b>. This option will display the panel's current connection status on the selected element.</p> <p><b>Notes</b></p> <ul style="list-style-type: none"> <li>The combination of Address Port and Address Code must be unique.</li> </ul> <p>The Address Port and Code assignments for Sub-Pages and Sub-Page View Buttons are provided only for use in SEND-COMMANDS (not to trigger actions).</p>

## State Properties

Listview Button-Specific State Properties	
<b>Secondary Font</b>	Use this property to specify the font used for the Secondary Font component (the second text line) of a Listview item.
<b>Secondary Font Size</b>	<p>Use this property to specify the font size used for the Secondary Font component (the second text line) of a Listview item.</p> <p>Note that the <i>Secondary Font</i> and <i>Secondary Font Size</i> State properties are available even if the selected Listview button only uses a single line of text. In this case, if the List View Type is changed to either two-line text or two-line text with icon, the second line of text will use these settings.</p>





### **Increase Your Revenue through education + knowledge**

In the ever-changing AV industry, continual education is key to success. AMX University is dedicated to ensuring that you have the opportunity to gather the information and experience you need to deliver strong AMX solutions. Plus, AMX courses also help you earn CEDIA, NSCA, InfoComm, and AMX continuing education units (CEUs).

Visit AMX University online for 24/7/365 access to:

- *Schedules and registration for any AMX University course*
- *Travel and hotel information*
- *Your individual certification requirements and progress*