Grandstream XML Application Guide

Three XML Applications

PART A – Application Explanations

PART B – XML Syntax, Technical Detail, File Examples

Grandstream XML Application Guide - PART A

Three XML Applications

Grandstream GXP Series supports both simple and advanced XML applications. Part A of this application note will describe Grandstream's three XML applications: 1) XML Custom Screen, 2) XML Downloadable Phonebook and 3) Advanced XML Survey Application.

Three XML Applications

Grandstream GXP Series supports both simple and advanced XML applications: 1) XML Custom Screen, 2) XML Downloadable Phonebook and 3) Advanced XML Survey Application.

WHAT IS XML?

XML is a markup language* for documents and applications containing structured information. This information contains both content (words, pictures, etc.) and an indication of what role that content plays (e.g. content in a section heading is different than content in a footnote, which is different than content in a figure caption or content in a database table, etc.). Almost all documents have some structure.

*A markup language is a mechanism to identify structures in a document. The XML specification defines a standard way to add markup to documents.

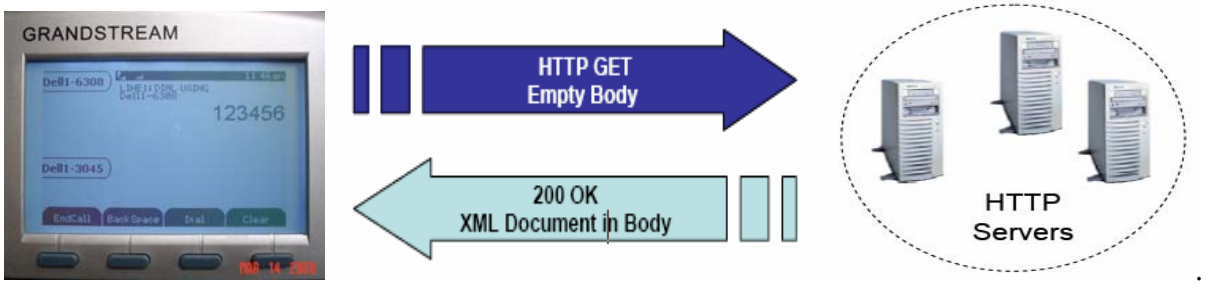
WHY XML?

What benefits does XML provide to SIP endpoints? XML enables our SIP phones to serve as output devices for many exciting applications. The XML infrastructure allows our phones to interact with external applications in a flexible and programmable manner. Three specific XML applications supported by Grandstream include *XML Custom Screen*, *XML Phonebook*, and *XML Survey Application*. The last application is a custom application where the XML framework is an interactive, real-time implementation and XML messaging is dynamic, depending on a configurable object set.

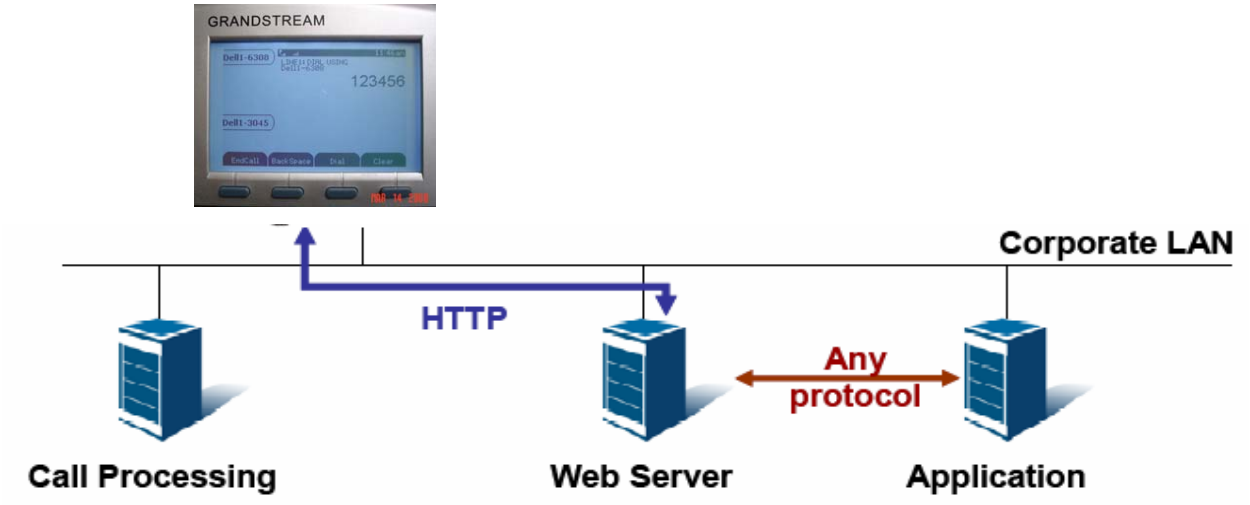
XML ARCHITECTURE

XML applications can be initiated in several ways.

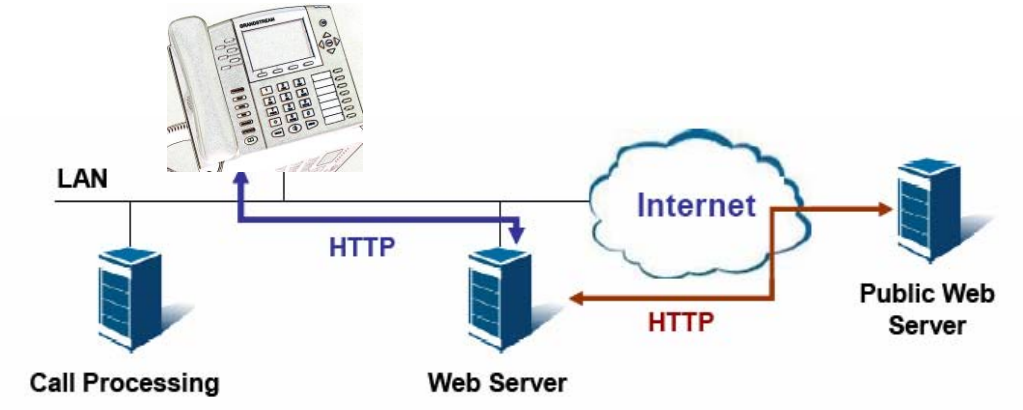
1. The first architecture is if the phone initiates a request for a service. In this instance, the GXP will receive a 200 OK with a XML document as its content.



2. The second architecture is in a closed network. An application in a closed network may exchange information in the following manner:



3. A third architecture enables the internal web server to interact with outside web server via HTTP.



As illustrated above, all of the application logic lies within the server side of the architecture. This allows faster applications development and minimal phone side maintenance. Users may develop *customized applications* using this infrastructure.

XML CUSTOM SCREEN APPLICATION

The GXP Series supports idle screen customization. The design and layout of the LCD screen is customer dependent. The following API configuration will work with all GXP Series models.

CUSTOM SCREEN API CONFIGURATION

The XML Custom Screen API is configurable in 2 steps:

1. **Enable Idle Screen XML Download** (P340): NO/YES-HTTP/YES-TFTP (default NO). Possible values 0 (NO)/1 (HTTP)/2 (TFTP), other values ignored.
2. **Idle Screen XML Server Path** (P341): This is a string of up to *128 characters* that contains a path to the XML file. It must follow the host/path format. **i.e.** "directory.grandstream.com/engineering"

The XML feature is activated when "Enable Idle Screen XML Download" is set to **YES** (HTTP or TFTP) and a valid "Idle Screen XML Server Path" is set.

To download or erase the XML file, the following 2 options should be selected in the ***Preference LCD GUI*** submenu:

- Download SCR XML
- Erase Custom SCR

Select the *Download SCR XML* using the Preference Keypad menu to start the download process. The phone will download the *gs_screen.xml* file specified in "Idle Screen XML Server Path". The XML application is effective immediately after download. Save the file for future use.

NOTE: the Idle Screen file name is fixed and cannot be changed based on personal preference.

XML PHONEBOOK APPLICATION

The GXP Series supports a downloadable phonebook with up to 500 entries. This application can also be used in large organizations where there is a central directory server so employees can efficiently download the most current directories.
NOTE: As of April 2008 Grandstream Networks is working on a solution to be able to download phonebooks that are 100+ entries large. The phones currently only support 500 entries if they are entered manually.
As a recommendation we suggest all XML phonebooks to be limited to 100 entries for the time being.

The following Phonebook API configuration will work with all GXP Series models.

PHONEBOOK API CONFIGURATION

The XML Custom Screen API is configurable in 4 steps:

1. **Enable Downloadable Phonebook (P330):** NO/YES-HTTP/YES-TFTP (default NO). Possible values 0 (NO)/1 (HTTP)/2 (TFTP), other values ignored.
2. **Phonebook XML Path (P331):** This is a string of up to *128 characters* that contains a path to the XML file. It must follow the host/path format. **i.e.** "directory.grandstream.com/engineering"
3. **Phonebook Download Interval (P332):** This field sets the time interval to download the phonebook (in hours) automatically. Valid value range is 0-720 (default 0).
4. **Remove manually remove edited entries on download:** YES/NO (default NO) P333, possible values 0/1, other values ignored.

The XML phonebook feature is activated when "*Enable Downloadable Phonebook*" is set to **YES** (HTTP or TFTP) and a valid "*Phonebook XML Path*" is set.

The phone will download the *gs_phonebook.xml* file specified in "*Phonebook XML Path*". The XML application is effective immediately after download. Save the file for future use. During this process the LCD will display a message to indicate the XML Phonebook download is in progress.

NOTE:

- The phonebook file name is fixed and cannot be changed based on personal preference.
- If the "*Phonebook Download Interval*" is set to a non-zero value *x*, the phonebook is automatically updated every *x* hours.
- If the "*Remove manually edited entries on download*" option is set to **No** (by default), the phone will keep ALL previously stored phonebook entries, insert the downloaded phonebook entries, and then save the phonebook. If set to **Yes**, the downloaded phonebook entries will replace the existing phonebook stored on the phone.
- At any time, you can immediately download the phonebook by choosing the "**Download Phonebook**" in the GUI Phone Book Menu (you can use the down arrow key when the phone is on-hook).

Grandstream XML Application Guide - PART B

XML Syntax, Technical Detail, File Examples

Part B of this application note will guide you through the technical details of our three applications: 1) XML Custom Screen, 2) XML Downloadable Phonebook and 3) Advanced XML Survey Application.

Application One: XML Custom Screen Details

1. XML Custom Screen Syntax

XSD file

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="Screen">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="IdleScreen" minOccurs="1" maxOccurs="1">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="ShowStatusLine" type="xsd:boolean" minOccurs="1" maxOccurs="1"
                                                                    default="true"/>
       <xsd:element name="DisplayBitmap" minOccurs="0" maxOccurs="unbounded" nillable="true">
        <xsd:complexType>
         <xsd:sequence>
          <!-- We only accept Windows Monochrome Bitmap, max 130x64 pixels encoded by base64 -->
          <xsd:element name="Bitmap" type="xsd:base64Binary" minOccurs="1" maxOccurs="1"/>
          <xsd:element name="X" type="xsd:integer" minOccurs="1" maxOccurs="1" default="0"/>
          <xsd:element name="Y" type="xsd:integer" minOccurs="1" maxOccurs="1" default="0"/>
         </xsd:sequence>
         <xsd:attribute name="a1reg" type="xsd:boolean"/>
        </xsd:complexType>
       </xsd:element>
       <xsd:element name="DisplayString" minOccurs="0" maxOccurs="unbounded" nillable="true">
        <xsd:complexType>
         <xsd:sequence>
          <xsd:element name="DisplayStr" type="xsd:string" minOccurs="1" maxOccurs="1"/>
          <xsd:element name="X" type="xsd:integer" minOccurs="1" maxOccurs="1" default="0"/>
          <xsd:element name="Y" type="xsd:integer" minOccurs="1" maxOccurs="1" default="0"/>
         </xsd:sequence>
         <xsd:attribute name="a1reg" type="xsd:boolean"/>
         <xsd:attribute name="font">
          <xsd:simpleType>
           <xsd:restriction base="xsd:string">
            <xsd:enumeration value="f8"/>
            <xsd:enumeration value="f10"/>
            <xsd:enumeration value="f13h"/>
            <xsd:enumeration value="f13b"/>
            <xsd:enumeration value="f16"/>
            <xsd:enumeration value="f16b"/>
            <!-- f18c is a 18 point Comic font  -->
            <xsd:enumeration value="f18c"/>
           </xsd:restriction>
          </xsd:simpleType>
         </xsd:attribute>
         <xsd:attribute name="halign">
          <xsd:simpleType>
           <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Left"/>
            <xsd:enumeration value="Center"/>
            <xsd:enumeration value="Right"/>
           </xsd:restriction>
          </xsd:simpleType>
         </xsd:attribute>
         <xsd:attribute name="valign">
          <xsd:simpleType>
           <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Top"/>
            <xsd:enumeration value="Center"/>
            <xsd:enumeration value="Bottom"/>
           </xsd:restriction>
          </xsd:simpleType>
         </xsd:attribute>
        </xsd:complexType>
       </xsd:element>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

2. Example Idle Screen File

```
<?xml version="1.0"?>
<!-- This file creates identical result to GXP-2000 default behavior -->
<Screen>
        <IdleScreen>
          <ShowStatusLine>true</ShowStatusLine>
          <DisplayString font="f8">
                <DisplayStr>$W, $M $d</DisplayStr>
                <X>0</X>
                <Y>0</Y>
          </DisplayString>
          <DisplayString font="f13h" halign="Center" a1reg="false">
                <DisplayStr>$N</DisplayStr>
                <X>65</X>
                <Y>12</Y>
          </DisplayString>
          <DisplayString font="f13b" halign="Center" a1reg="true">
                <DisplayStr>$N</DisplayStr>
                <X>65</X>
                <Y>12</Y>
          </DisplayString>
          <DisplayString font="f13h" halign="Center" a1reg="false">
                <DisplayStr>$X</DisplayStr>
                <X>65</X>
                <Y>26</Y>
          </DisplayString>
          <DisplayString font="f13b" halign="Center" a1reg="true">
                <DisplayStr>$X</DisplayStr>
                <X>65</X>
                <Y>26</Y>
          </DisplayString>
          <DisplayString halign="Center" valign="Bottom">
                <DisplayStr>$I</DisplayStr>
                <X>65</X>
                <Y>48</Y>
          </DisplayString>
        </IdleScreen>
</Screen>
```

NOTE: As of April 2008 Grandstream Networks is working on a solution regarding the loss of softkey functionality when the phone loads a custom idle screen. This means that if the user chooses to display his own idle screen he will loose the softkey functions until we release a solution.

3. XML Explanation

ROOT ELEMENT "SCREEN"

The XML document has root element called `Screen`; it contains exactly 1 sub-element called `IdleScreen`.

```
<xsd:element name="Screen">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="IdleScreen" type="IdleScreenType"
                   minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

ELEMENT "IDLESCREENTYPE"

This element defines three components that are makes up the idle screen. These components are defined as elements.

```
<xsd:complexType name="IdleScreenType">
  <xsd:sequence>
    <xsd:element name="ShowStatusLine" type="xsd:boolean"
                 minOccurs="1" maxOccurs="1" default="true"/>
    <xsd:element name="DisplayBitmap" type="DisplayBitmapType"
                 minOccurs="0" nillable="true"/>
    <xsd:element name="DisplayString" type="DisplayStringType"
                 minOccurs="0" nillable="true"/>
  </xsd:sequence>
</xsd:complexType>
```

Note: ShowStatusLine must appear exactly once and any number of DisplayBitmap and DisplayString instances.

DISPLAY RULES

When both DisplayBitmap and DisplayString elements are present, all bitmaps will be rendered before the strings are displayed. When multiple instances of the same type (bitmap/string) are present, they are displayed in the order they appear in the XML and later objects (bitmap/string) may overwrite/corrupt previous objects.

ELEMENT "SHOWSTATUSLINE"

This Boolean element displays the status bar on the top of the screen. The "Status Line" includes the registration status icon, volume icon, time/date on the right-top corner, and the horizontal separator line. This element must appear exactly once in *IdleScreenType* and has a default value of "true". When set to false, the origin (x-0, y-0) refers to the absolute top-left corner; when set to true, the origin refers to the reference-origin below status line (x-0,y-16). This means when *ShowStatusLine* is set to true, all y-offsets will shift down 16 pixels so the status line will not be corrupted or over-written.

ELEMENT DISPLAYBITMAP

This element carries the information on how a bitmap is to be rendered on screen. It has three mandatory elements and one optional attribute:

Element Bitmap

This element contains the bitmap encoded by base64

```
<xsd:element name="Bitmap" type="xsd:base64Binary" minOccurs="1" maxOccurs="1"/>
```

Note: The file must be a Windows Bitmap (file header begins with 0x424D) that is monochrome (1-bit depth) and not exceeding 130x65 pixels (that's our LCD resolution). Anything not bound to the above restriction is dropped and ignored. You may use Windows Paint to change an existing BMP file to 1-bit depth.

Elements X and Y

This element contains X and Y offsets from the origin that we will use to render the bitmap.

```
<xsd:element name="X" type="xsd:integer" minOccurs="1" maxOccurs="1" default="0"/>
<xsd:element name="Y" type="xsd:integer" minOccurs="1" maxOccurs="1" default="0"/>
```

Attribute a1reg

This OPTIONAL attribute specifies the conditions when the bitmap will be displayed.

```
<xsd:attribute name="a1reg" type="xsd:boolean"/>
```

When this attribute is present and the value is "true" then the bitmap will be displayed ONLY when SIP Account 1 is in REGISTERED state.

When this attribute is present and the value is "false" then the bitmap will be displayed only when SIP Account 1 is NOT in REGISTERED state.

Note: When this attribute is absent then this bitmap is displayed regardless to the SIP Account 1 registration states.

ELEMENT DISPLAYSTRING

This element carries the information on how a string is to be rendered on screen. It has three mandatory elements and four optional attributes:

Element DisplayStr

This element contains the string to be displayed

```
<xsd:element name="DisplayStr" type="xsd:string" minOccurs="1" maxOccurs="1"/>
```

The string can contain dynamic contents. As present, we support the following 19 system variables that will be substituted with dynamic contents at run-time.

1. $W: This variable is replaced with the current day of week and has the following possible values: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday
2. $N: This variable is replaced with the configured Account 1 Display Name.
3. $X: This variable is replaced with the configured Account 1 SIP User ID.
4. $V: This variable is replaced with the configured Account 1 SIP Server.
5. $I: This variable is replaced with the system IP address.
6. $D: This variable is replaced with the current day of month with leading zero, possible values: 01, 02, …, 31
7. $d: This variable is replaced with the current day of month without leading zero, possible values: 1, 2, …, 31
8. $M: This variable is replaced with the current month in English, possible values: January, February, …, December
9. $o: This variable is replaced with the current month in number with leading zero, possible values: 01, 02, …, 12
10. $n: This variable is replaced with the current month in number without leading zero, possible values: 1, 2, …, 12
11. $Y: This variable is replaced with the current year in 4-digit number, for example: 2006, 2007 …
12. $y: This variable is replaced with the current year in 2-digit number, for example: 06, 07 …
13. $P: This variable is replaced with the current AM/PM status in upper case, possible values: AM, PM
14. $p: This variable is replaced with the current AM/PM status in lower case, possible values: am, pm
15. $H: This variable is replaced with the current hour of day in 24-hour representation with leading zero, possible values: 00, 02, …, 23
16. $h: This variable is replaced with the current hour of day in 12-hour representation with leading zero, possible values: 01, 02, …, 12
17. $m: This variable is replaced with the current minute of hour with leading zero, possible values: 01, 02, …, 59
18. $s: This variable is replaced with the current second of minute with leading zero, possible values: 01, 02, …, 59
19. $i: This variable is replaced with the current number of missed calls on the phone, possible values: 01, 02, 03…50.

Note: If you want to display the "$" sign, use the "$$" escape sequence.

Element displayCondition

This element contains a Boolean condition (true or false) that will trigger the display of a string upon an event. This element needs a condition type to be specified for each condition that we want to change, for example:

<displayCondition><conditionType>dnd</conditionType></displayCondition>

Will display a string associated with the dnd condition when the phone is set to Do Not Disturb. So in the XML file we would have the following:

```
<DisplayString font="f13b" halign="Right" >
        <DisplayStr>DND</DisplayStr>
        <X>80</X>
        <Y>80</Y>

    <displayCondition><conditionType>dnd</conditionType></displayCondition>
</DisplayString>
```

This section will display the string "DND" in the lower portion of the screen upon the user activating the DND function on the phone.
In a similar manner the following condition types are supported:

1. dnd: This condition type will be displayed when the phone is set to Do Not Disturb.
2. misCall: This condition type will be displayed when the phone logs any missed calls.
3. callFwded: This condition type will be displayed when the user activates call forward via the keypad dialing *72, (*73 will deactivate this feature.)
4. a#reg: Where # could be any number from 1 to 6 depending on the GXP model. This condition type will be displayed if account # is registered.
5. a#voiceMsg: Where # could be any number from 1 to 6 depending on the GXP model. This condition type will de displayed when account # has a voice message in the mailbox.

Elements X and Y

This element contains X and Y offsets from the origin that we will use to render the string.

```
<xsd:element name="X" type="xsd:integer" minOccurs="1" maxOccurs="1" default="0"/>
<xsd:element name="Y" type="xsd:integer" minOccurs="1" maxOccurs="1" default="0"/>
```

Attribute a1reg

This OPTIONAL attribute specifies the conditions when the string will be displayed.

```
<xsd:attribute name="a1reg" type="xsd:boolean"/>
```

When this attribute is present and the value is "**true**" then the string will be displayed ONLY when SIP Account 1 is in REGISTERED state. When this attribute is present and the value is "**false**" then the string will be displayed only when SIP Account 1 is NOT in REGISTERED state.

Note: When this attribute is absent default is this bitmap is displayed regardless of the SIP Account 1 registration.

Attribute font

This OPTIONAL attribute specifies the font we will use to render the string.

```
<xsd:attribute name="font" type="fontType"/>

<xsd:simpleType name="fontType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="f8"/>
      <xsd:enumeration value="f10"/>
      <xsd:enumeration value="f13h"/>
      <xsd:enumeration value="f13b"/>
```

```
            <xsd:enumeration value="f16"/>
            <xsd:enumeration value="f16b"/>
            <!-- f18c is a 18 point Comic font  -->
            <xsd:enumeration value="f18c"/>
          </xsd:restriction>
      </xsd:simpleType>
```

This application supports 1 system font and 7 additional fonts in various sizes as enumerated above. Any fonts not recognize will default to the system font. When this attribute is absent, default is system font.

Attribute halign

This OPTIONAL attribute specifies the horizontal alignment method used to display the string.

```
      <xsd:attribute name="halign" type="HorizontalAlignmentType"/>

      <xsd:simpleType name="HorizontalAlignmentType">
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Left"/>
            <xsd:enumeration value="Center"/>
            <xsd:enumeration value="Right"/>
          </xsd:restriction>
      </xsd:simpleType>
```

Default is to Left when this attribute is absent.

Note: When using the Center alignment you will need to calculate the midpoint for the x-coordinate (4.5.2) for the width to be considered centered. For instance, element X must be set to 65 (130/2) to display a string that is aligned to the center of the LCD. Similarly, you will need to specify the right most point to render if you are using the Right halign method.

Attribute valign

This OPTIONAL attribute specifies the vertical alignment method used to display the string.

```
      <xsd:attribute name="valign" type="VerticalAlignmentType"/>

      <xsd:simpleType name="VerticalAlignmentType">
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Top"/>
            <xsd:enumeration value="Center"/>
            <xsd:enumeration value="Bottom"/>
          </xsd:restriction>
      </xsd:simpleType>
```

Default is to Top when this attribute is absent.

Note: When using the Center alignment, calculate the midpoint for the y-coordinate (4.5.3) in order for the width to be centered. **i.e.** element Y must be set to 32 to display a string that is aligned to the center of the LCD. Similarly, you will need to specify the right most point to render if you are using the Bottom valign method (set to 64).

Application 2: XML Phonebook Details

1. XML Phonebook Syntax

XSD file

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="AddressBook">
    <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="Contact" minOccurs="1" maxOccurs="unbounded">
         <xsd:complexType>
          <xsd:sequence>
```

```
            <xsd:element name="LastName" type="xsd:string" minOccurs="1"/>
            <xsd:element name="FirstName" minOccurs="0" type="xsd:string" nillable="true"/>
            <xsd:element name="Address" minOccurs="0" nillable="true">
             <xsd:complexType>
              <xsd:sequence>
               <xsd:element name="address1" type="xsd:string" minOccurs="1" maxOccurs="1"/>
               <xsd:element name="address2" type="xsd:string" minOccurs="0" maxOccurs="1"/>
               <xsd:element name="city" type="xsd:string" minOccurs="1" maxOccurs="1"/>
               <xsd:element name="state" type="xsd:string" minOccurs="1" maxOccurs="1"/>
               <xsd:element name="zipcode" type="xsd:string" minOccurs="1" maxOccurs="1"/>
               <xsd:element name="country" type="xsd:string" minOccurs="1" maxOccurs="1"/>
              </xsd:sequence>
             </xsd:complexType>
            </xsd:element>
            <xsd:element name="Phone">
             <xsd:complexType>
              <xsd:sequence>
               <xsd:element name="phonenumber" type="xsd:string" minOccurs="1" maxOccurs="1"/>
               <xsd:element name="accountindex" type="xsd:integer" minOccurs="1" maxOccurs="1"
                                                                                  default="1"/>
              </xsd:sequence>
             </xsd:complexType>
            </xsd:element>
            <xsd:element name="Email" type="xsd:string" minOccurs="0" nillable="true"/>
            <xsd:element name="Department" type="xsd:string" minOccurs="0" nillable="true"/>
            <xsd:element name="Company" type="xsd:string" minOccurs="0" nillable="true"/>
            <xsd:element name="Icon" type="xsd:base64Binary" minOccurs="0" nillable="true"/>
           </xsd:sequence>
          </xsd:complexType>
         </xsd:element>
        </xsd:sequence>
       </xsd:complexType>
      </xsd:element>
</xsd:schema>
```

2. Example XML Phonebook File

```
<?xml version="1.0"?>
<AddressBook>
        <Contact>
          <LastName>Doe</LastName>
          <FirstName>John</FirstName>
          <Phone>
            <phonenumber>8000</phonenumber>
            <accountindex>0</accountindex>
          </Phone>
        </Contact>
        <Contact>
          <LastName>Smith</LastName>
          <FirstName>Alan</FirstName>
          <Phone>
            <phonenumber>8001</phonenumber>
            <accountindex>0</accountindex>
          </Phone>
        </Contact>
        <Contact>
          <LastName>Lee</LastName>
          <FirstName>Lily</FirstName>
          <Phone>
            <phonenumber>6000</phonenumber>
            <accountindex>1</accountindex>
          </Phone>
        </Contact>
</AddressBook>
```

Application Three: XML Call Center/Survey Details

XML CALL CENTER / SURVEY DETAILS

1. XML Phonebook Syntax

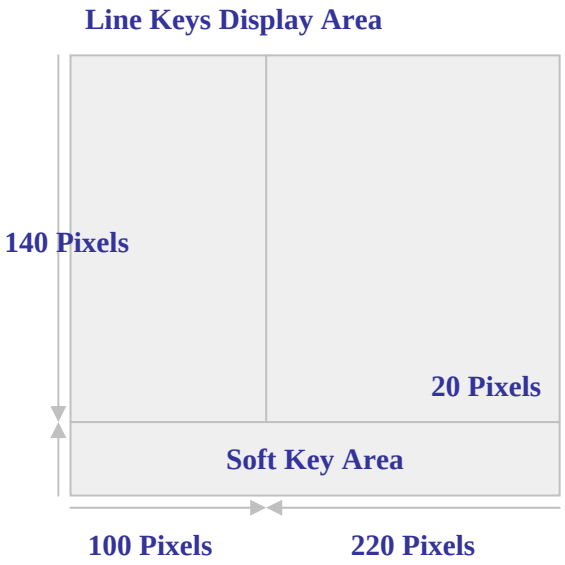
This section gives a brief overview of the screen layout and the XML syntax.

Special Characters

As followed by the standard XML recommendation, some characters need to be escaped. The following list the characters together with their escape sequence:

Characters	Name	Escape Sequence
&	Ampersand	&amp
“	Quote	&quot
‘	Apostrophe	&apos
<	Left Angle Bracket	&lt;
>	Right Angle Bracket	&gt;

GRANDSTREAM XML OBJECT



GXP2020 XML Display

As illustrated above, XML content will be displayed in the 140 X 220 display area. Accordingly, the soft keys will be displayed in their respective area. A single XML object will represent the display area. This XML object takes precedence over the current display. It will also overwrite the soft keys functionality to give users control over the XML.

Grandstream GS_XML Object Descriptions

Name	Location	Type	Values	Comments
GS_XML_Application	Root	Mandatory	-	Root element
Display	Root Body	Optional	-	Screen display
SoftKeys	Root Body	Optional	-	Programmable Softkey
Events	Root Body	Optional	-	Determines what action to be

	done when a local event is detected

Display Object Description

Name	Location	Type	Values	Comments
Display	Root Body	Optional	-	Screen display
Screen	Display Body	Display Choice	-	Unselectable display
*Menu	Display Body	Display Choice	-	Selectable display

***** indicates future implementation.

Screen Object Descriptions

Name	Location	Type	Values	Comments
Screen	Display Body	Display Choice	-	Unselectable display
showLineLabels	Screen Attribute	Optional	Boolean	Enable or disable line labels display to extend xml application display area
DisplayString	Screen Body	Optional	-	Display a string on the screen
DisplayBitmap	Screen Body	Optional	-	Display a bitmap on the screen

DisplayBitmap Object XML Description

Name	Location	Type	Values	Comments
DisplayBitmap	Screen Body	Optional	-	Display a bitmap on the screen
X	DisplayBitmap Body	Mandatory	Integer	Horizontal starting position depending on halign
Y	DisplayBitmap Body	Mandatory	Integer	Vertical starting position depending on valign
Bitmap	DisplayBitmap Body	Mandatory	Strings	Content of the bitmap

DisplayString Object XML Description

Name	Location	Type	Values	Comments
DisplayString	Screen Body	Optional	-	Descriptions of a

				line string
Font	DisplayString Attribute	Optional	Enum string	The font for a line
Halign	DisplayString Attribute	Optional	Left, Center, Right	Horizontal Axis Alignment
Valign	DisplayString Attribute	Optional	Top, Center, Bottom	Vertical Axis Alignment
Wrap	DisplayString Attribute	Optional	Boolean	Wrap the text to the next line
X	DisplayString Body	Mandatory	Integer	Horizontal starting position depending on halign
Y	DisplayString Body	Mandatory	Integer	Vertical starting position depending on valign
DisplayStr	DisplayString Body	Mandatory	String	A String to display

Softkeys Object XML Description

Name	Location	Type	Values	Comments
SoftKey	Display Body	Optional	-	Programmable Softkey
Label	Softkey body	Mandatory	String	Label for the index
Action	Softkey Body	Mandatory	Enum String	Local Action

Action Object XML Description

Name	Location	Type	Values	Comments
Action	Softkey Body	Mandatory	-	Action to be taken
UseURL	Action body	Choice	-	URL to call to
QuitApp	Action Body	Choice	-	Quit the current XML application
Dial	Action Body	Choice	-	Dial a number

The remote server may specify actions for the local phone. These actions must be pre-defined *locally* before they can be executed. Currently, the local action is to quit the current XML application.

Dial Object XML Description

Name	Location	Type	Values	Comments
Dial	Action Body	Choice	-	Dial a number
Account	Dial body	Mandatory	Integer	Account to be used
Number	Dial Body	Mandatory	Integer	Number to dial

Events Object XML Description

Name	Location	Type	Values	Comments
Event	Events body	Optional	-	Describe a specific event

Event Object XML Description

Name	Location	Type	Values	Comments
State	Event Body	Mandatory	Enumerated string	The possible state of the phone
Action	Event Body	Mandatory	-	Action to be taken if phone changes to that state

An action is triggered only if the phone **transition INTO** such state.

2. EXAMPLE OF SURVEY IMPLEMENTATION

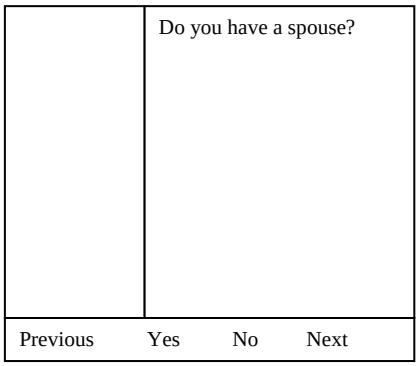
An initial URI is configurable using the web interface. This is the location of the server that the phone will attempt to interact with. It can be triggered by, lets say, overloading the phonebook button or the right arrow button or even by providing a menu in which the user can select with from the phone. In this example, we will use the initial URL to be: http://www.XMLquestionnaire.com:123 where 123 represent the port.

Once the XML service is triggered, the phone will send an empty HTTP GET request to the above URL at the selected port. The server will respond with a 200 OK with the XML attached to the packet. The XML syntax may look like this:

```
<GS_XML_Application>
        <Display>
                <Screen>
                        <DisplayString>
                        <X>0</X>
                        <Y>0</Y>
                        <DisplayStr> Do you have a spouse?</DisplayStr>
                        </DisplayString>
                </Screen>
        </Display>
        <SoftKeys>
                <SoftKey>
                        <Label>Previous</Label>
                        <Action>
                                <UseURL>
                                <URL>http://www.XMLquestionnaire.com/questionnaire.php?do=prev
</URL>
                                </UseURL>
                        </Action>
                </SoftKey>
                <SoftKey>
                        <Label>Yes</Label>
                        <Action>
                                <UseURL>
                                <URL>http://www.XMLquestionnaire.com/questionnaire.php?ans=yes
</URL>
                                </UseURL>
                        </Action>
                </SoftKey>
```

```
                <SoftKey>
                        <Label>No</Label>
                        <Action>
                                <UseURL>
                                <URL>http://www.XMLquestionnaire.com/questionnaire.php?ans=no </URL>
                                </UseURL>
                        </Action>

                </SoftKey>
                <SoftKey>
                        <Label>Next</Label>
                        <Action>

                                <UseURL>
                                <URL>http://www.XMLquestionnaire.com/questionnaire.php?do=next
</URL>
                                </UseURL>
                        </Action>
                </SoftKey>
        </SoftKeys>
</GS_XML_Application >
```

After receiving the 200 OK, the display and softkey area of the screen is changed to show the XML content:

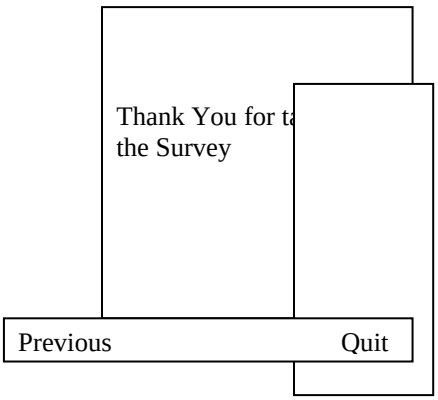


Each softkey is bound by a URI, and pressing the soft key will call the corresponding URI. If the second softkey is pressed http://www.XMLquestionnaire.com/questionnaire.php?ans=yes is called. The server now knows the client response because the URI given to each soft key is unique. Once the answer is received, the server will send a 200 ok with another XML attached.

```
<GS_XML_Application>
        <Display>
                <Screen>
                        <DisplayString valign="Center">
                                <X>0</X>
                                <Y>0</Y>
                                <DisplayStr> Thank you for taking the</DisplayStr>
                         </DisplayString >
                        < DisplayString >
                        <X>0</X>
                        <Y>10</Y>
                        <DisplayStr>Survey</DisplayStr>
                         </DisplayString >
                        <DisplayString halign="Right" valign="Bottom">
```

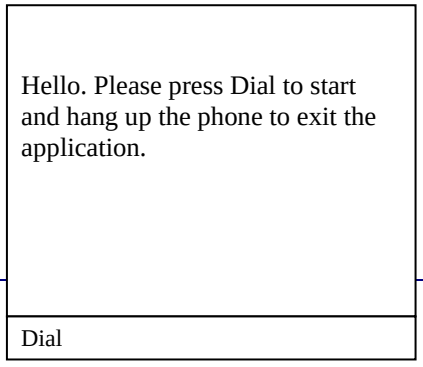
```
                <X>0</X>
                <Y>0</Y>
                <DisplayStr>Surveyor</DisplayStr>
                </ DisplayString>
            </Screen>
        </Display>

        <SoftKeys>
            <SoftKey>
                <Label>Previous</Label>
                <Action>
                <UseURL>
                    <URL>http://www.XMLquestionnaire.com/questionnaire.php?do=prev
                </URL>
                </UseURL>
                </Action>

            </SoftKey>
            <SoftKey>
                <Label>Quit</Label>
                <Action>
                    <QuitApp/>
                 </Action>
            <SoftKey>
        </SoftKeys>
</GS_XML_Application>
```

In this XML, note that the last word, “Survey”, displayed at the center of the screen is in a new Line element. Otherwise, it would be displayed off the screen and look like this screen:



EXAMPLE XML FILE

The following XML file syntax uses Dial action and Event elements and the Line Labels turned off. This syntax will display this introduction screen:



XML SYNTAX

```
<GS_XML_Application>
        <Display>
                <Screen showLineLabels="false">
                        <DisplayString>
                                <X>0</X>
                                <Y>0</Y>
                                <DisplayStr wrap = "true"> Hello. Please press Dial to start dialing and hang
                                up the phone to exit the application. </DisplayStr>
                        </DisplayString>

                </Screen>
        </Display>
<SoftKeys>
        <SoftKey>
                <Label>Dial</Label>
                <Action>
                        <Dial>
                                <Account>0</Account>
                                <Number>3051</Number>
                        </Dial>
                </Action>
        </SoftKey>
</SoftKeys>

<Events>
        <Event>
                <State>Onhook</State>
                <Action>
                        <QuitApp/>
                </Action>
        </Event>
</Events>

</GS_XML_Application>
```

Pressing **Dial** dials 3051 with Account 1. After the survey is finished and the user hangs up, the survey is ended. The QuitApp action is triggered.

XML SURVEY CONFIGURATION

1. Before the XML survey application can be launched, the GXP2020 firmware must be upgraded to version 1.1.5.x.
2. After upgrading and rebooting, the XML server path on the advanced settings tab must be set to point to the server where the application is located (i.e. 12.34.56.78/test_gs.xml). Please make note that the account 1 page must be configured with a valid SIP account for the application to run. Reboot the phone again.
3. When the phone boots up, the first soft key will be "XML Service". Press this key to get started. It is also possible to enter the application server path into a web browser. This way you'll be able to see on your screen the exact XML that your phone is receiving also. If you press a soft key on your phone, you'll just need to navigate to the URL of the soft key, as can be seen in the XML code. For example, on 12.34.56.78/test_gs.xml, you'll see that the first soft key is set to make the phone browse to 12.34.56.78/politics_gs_start.xml. Press that soft key and enter that URL into your web browser, and you'll be seeing the XML code on your screen and the actual displayed page on your phone. This makes the implementation of the application relatively easy to understand.
4. Once the application is launched the first question of the survey will appear on the main body of the GXP LCD screen. The user can then answer the multiple choice questions with the corresponding soft key. Note: In some cases you will have to click on the "more" softkey if there are more than 4 multiple choice answers.
5. After completing one section of the survey, the user can jump to another category or automatically go to the next category by pressing the corresponding softkey.
6. After completing all the survey category's the application will prompt the user to exit.