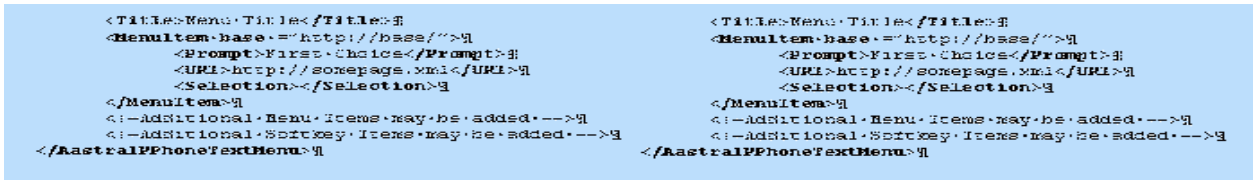
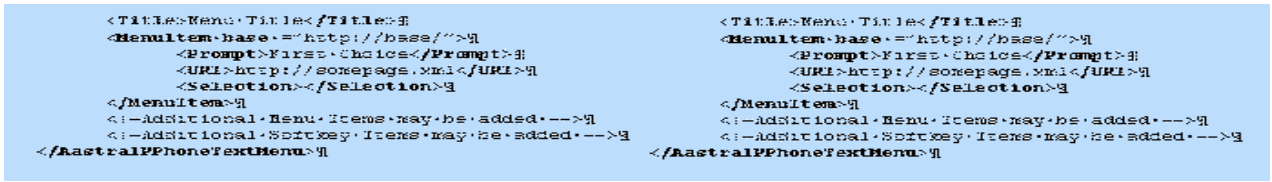


Development Guide
XML API for Aastra SIP Phones
Firmware 2.1.0

Doc. No.: PA-001008-00-00



Aastra Telecom will not accept liability for any damages and/or long distance charges, which result from unauthorized and/or unlawful use.

While every effort has been made to ensure accuracy, Aastra Telecom will not be liable for technical or editorial errors or omissions contained within this documentation. The information contained in this documentation is subject to change without notice.

TABLE OF CONTENTS

TABLE OF FIGURES

1 Introduction

1.1 Aastra XML browser

Aastra SIP phones have a XML browser embedded since firmware release 1.3.0. This XML browser allows external applications to control the display of the phone.

The list of XML applications is endless. See chapter 8 for some potential applications.

This document details the XML objects supported by the Aastra SIP Phones and how to implement them.

1.2 Revision History

1.2.1 Version 2.1.0 (53i/55i/57i/57i CT)

- "doneAction" root tag for the AastraIPPhoneTextScreen and AastraIPPhoneFormattedTextScreen XML objects. This tag allows redirecting the user to a specified URI after a "Done" key press; this can be very useful for non softkey phones such as the Aastra 53i.
- New custom softkeys "Ignore" and "Answer" and "allowAnswer" tag to answer a call when an XML page is called upon the incoming call.
- New custom softkey (List) for the AastraIPPhoneInputScreen to enter a list of configured symbols. This new softkey allows for instance an easy email address input using the "@." List of symbols.
- New configuration parameter `xml get timeout` added in order to control the server answer delay.
- New configuration parameters `services script` and `callers list script` to override internal applications and link the features to an XML script.

1.2.2 Version 2.0.2 (53i/55i/57i/57i CT)

- New command "FastReboot" for the PhoneExecute object to trigger a fast reboot of the phone (no firmware check and limited language package check).
- New universal URI type "Led:" support to control the LED state of the phone keys when they are typed as XML.
- Extension of the AastraIPPhoneInputScreen XML object to support multiple input fields (55i/57i and 57i CT only)

1.2.3 Version 2.0.1 (53i/55i/57i/57i CT)

<u>New XML objects</u>

- AastraIPPhoneConfiguration
- AastraIPPhoneImageScreen (55i/57i/57i CT)
- AastraIPPhoneImageMenu (55i/57i/57i CT)
- AastraIPPhoneFormattedScreen

<u>Other</u>

- HTTPS support for XML calls.

- Custom port support for http(s) XML calls
- “dial:XXXX” universal URI support
- New “style” tag for AastraIPPhoneTextMenu
- Optional “Title” tag for all UI objects
- “Title” wrapped on 2 lines (“wrap” tag).
- Cancel remap for all UI XML objects
- Timeout attribute common to all UI XML objects
- LockIn attribute common to all UI XML objects
- triggerDestroyOnExit for non UI objects
- Icons in TextMenu
- Icons in customizable softkeys (55i/57i/57i CT)
- New HTTP header to indicate the presence of expansion modules
- New input types for AastraIPPhoneInputScreen
    - timeUS, timeInt
    - dateUS, dateInt

1.2.4 Version 1.4.1 (9112i/9133i/480i/480i CT)

New XML objects

- AastraIPPhoneStatus
- AastraIPPhoneExecute

Action URIs

- End of the boot sequence — action uri startup
- Successful registration — action uri registered
- On-hook — action uri onhook
- Off-hook — action uri offhook
- Incoming call — action uri incoming
- Outgoing call — action uri outgoing

URI System variables

- $$SIPUSERNAME$$ — line user name
- $$DISPLAYNAME$$ — the display name of the focused line
- $$SIPAUTHNAME$$ — the SIP auth name of the focused line
- $$PROXYURL$$ — the SIP proxy of the focused line
- $$INCOMINGNAME$$ — returns the Caller-ID of the incoming call
- $$REMOTENUMBER$$ — returns the number of the remote party (incoming or outgoing)

Other

- No more need of URL encoding for pushed pages

- Beep attribute common to all XML objects

1.2.5 Version 1.3.1 (9112i/9133i/480i/480i CT)

Customizable softkeys (480i/480i CT)

- Select
- Exit
- Dial
- Submit
- BackSpace
- NextSpace
- Dot
- ChangeMode

Other

- XML softkeys (480i/480i CT)
- XML programmable keys (9112i/9133i)
- destroyOnExit attribute common to all XML objects

1.2.6 Version 1.3.0 (9112i/9133i/480i/480i CT)

Fist revision of this document

New XML objects

- AastraIPPhoneTextScreen
- AastraIPPhoneTextMenu
- AastraIPPhoneInputScreen
- AastraIPPhoneDirectory (480i/480i CT)

2 XML and the Aastra IP Phones

2.1 What is XML?

XML stands for **eXtensible Markup Language**. It is a markup language much like HTML. HTML was designed to display data and to focus on how data looks. XML was designed to describe data and to focus on what data is.

The following are characteristics of XML:

- XML tags are not predefined. You must define your own tags
- XML uses a Document Type Definition (DTD) or an XML Schema to describe the data
- XML with a DTD or XML Schema is designed to be self-descriptive
- XML is a W3C Standard Recommendation

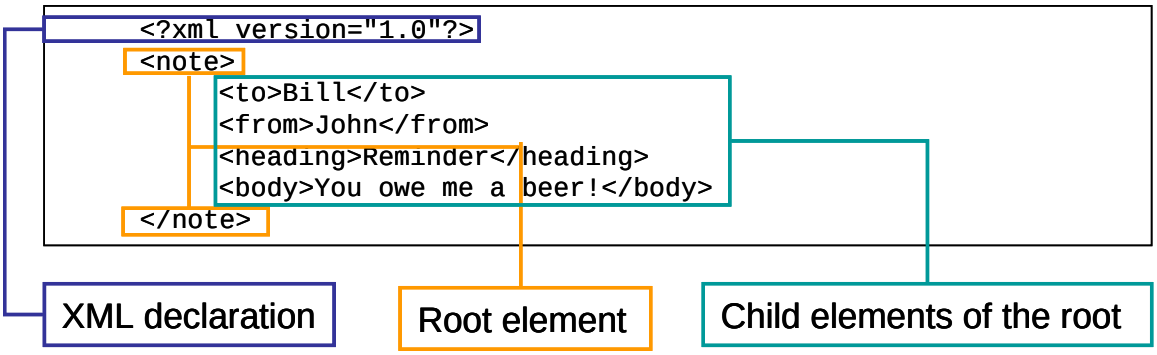


Figure 1: Basic XML document

More information available at http://www.xml.com

2.2 Functionality

The XML browser in Aastra IP phones allows developers to create custom services that they can use via the phone's keypad and display. These services include things like weather and traffic reports, contact information, company info, stock quotes, or custom call scripts.

With firmware release 2.0, the Aastra IP phone XML API supports 10 proprietary objects that allow the creation of powerful XML applications.

There are 2 types of XML objects:

- UI objects, XML objects which will use the display of the phone when they are received.
- Non UI objects

The supported objects are:

- TextMenu object
- ImageMenu object
- TextScreen object
- FormattedTextScreen object
- ImageScreen object
- InputScreen object
- Directory object
- Status object
- Execute object
- Configuration object

UI Objects

Non UI objects

Some of these objects also support customizable softkeys that are declared as an independent object (55i/57i/57i CT only).

The following sections describe the process of creating XML objects for the Aastra IP phones.

Note: the XML browser is operational only if the phone Web Server is enabled.

2.3 How does it work?

Leveraging on the IP infrastructure, Aastra has decided to develop the browser capability on the phone using the HTTP transport protocol but as a direct support of HTML would not be suitable for the phone horsepower and limited display, the choice has been made to support only XML objects in the browser.

The Aastra IP phones support two types of applications:

- Phone-initiated
- Server-initiated

2.3.1 Phone initiated application

The phone issues an HTTP (or HTTPS) GET command to the Web server, waits for the answer, decodes and displays this answer as any Web browser such as Microsoft Internet Explorer or Firefox would do as a Web client.

This can be done through a phone custom softkey and from the list of custom features (see chapter 7 for more details).

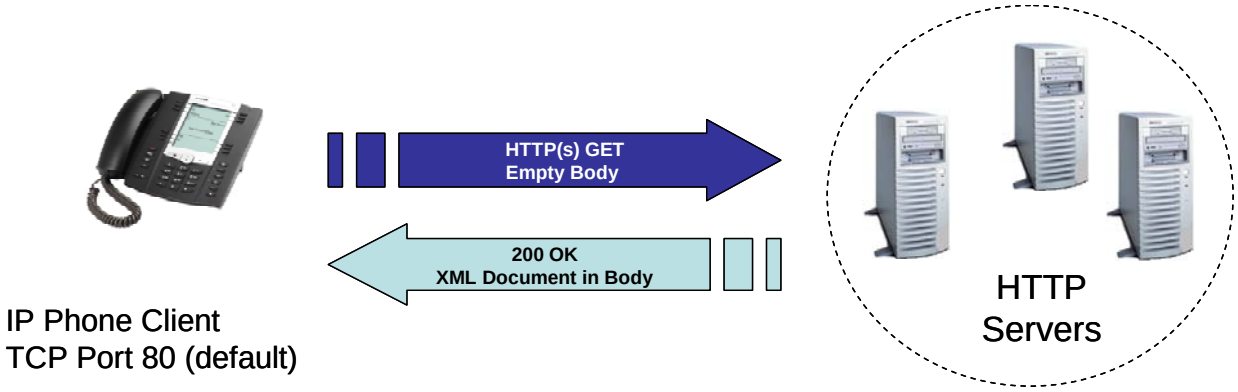


Figure 2: Aastra IP Phone acting as a client

2.3.2 Server initiated application

The other type of application would be more used for alerting as an application is pushing an XML object to the phone. The phone is now acting as a limited Web Server.

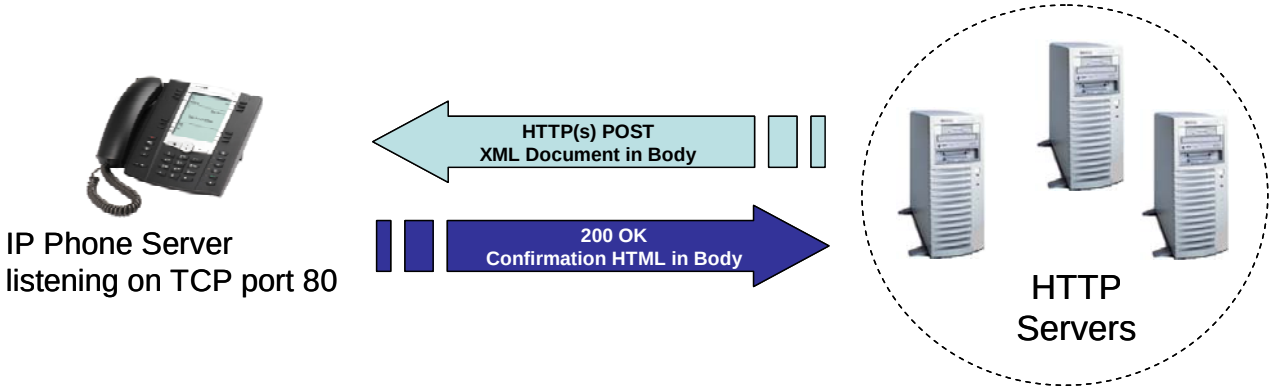


Figure 3: Aastra IP Phone acting as a server

2.4 System Architecture

The XML applications are hosted by one or multiple Web servers which will serve as a proxy to either other applications or to Internet Web servers.

2.4.1 Corporate applications

The following figure details the architecture to allow Aastra IP Phones to access an internal application. The application hosted by the Web server translates the phone requests to a protocol specific to the target application and formats the answer as an XML object to be displayed on the phone.

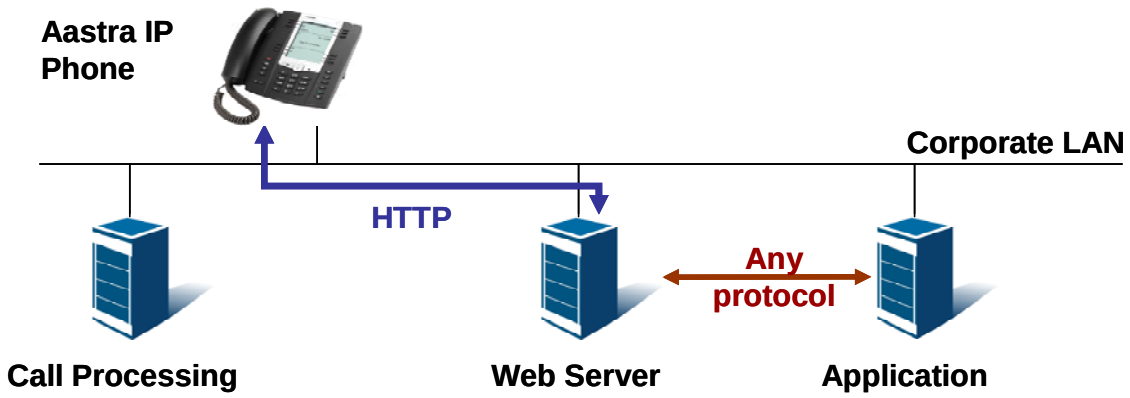


Figure 4: Access to an internal application

2.4.2 Internet applications

The following figure details the architecture of an XML application that would retrieve data from the internet such as a real-time stock-quote service.

Note: for certain Web applications that are not real time, the Internet content can be cached on the XML web server for a faster access.

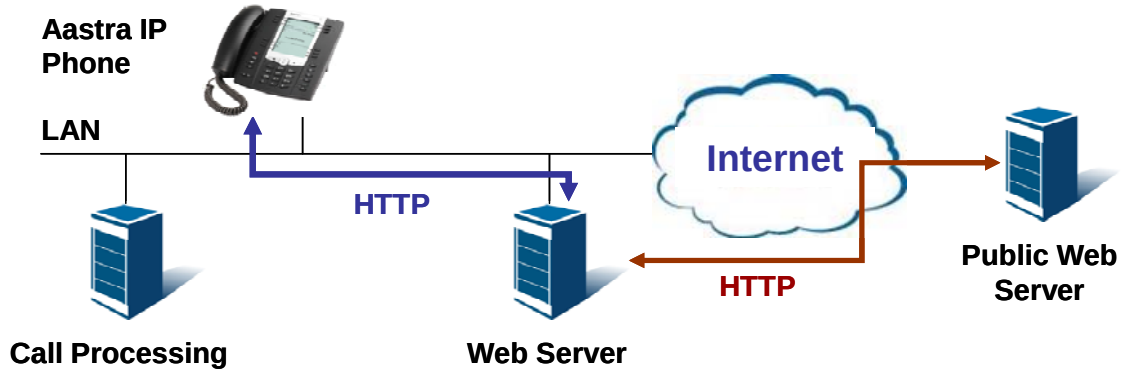


Figure 5: Access to an Internet application

2.4.3 Telephony applications

The following figure details the architecture of an XML application that would provide more telephony features at the phone level.

The application could for instance

- show the list of the parked calls and perform a pick-up,
- activate the Call forward or the DND on the server side
- control a conference from the phone
- Login/logout from a call center, access to the voice mail messages
- …

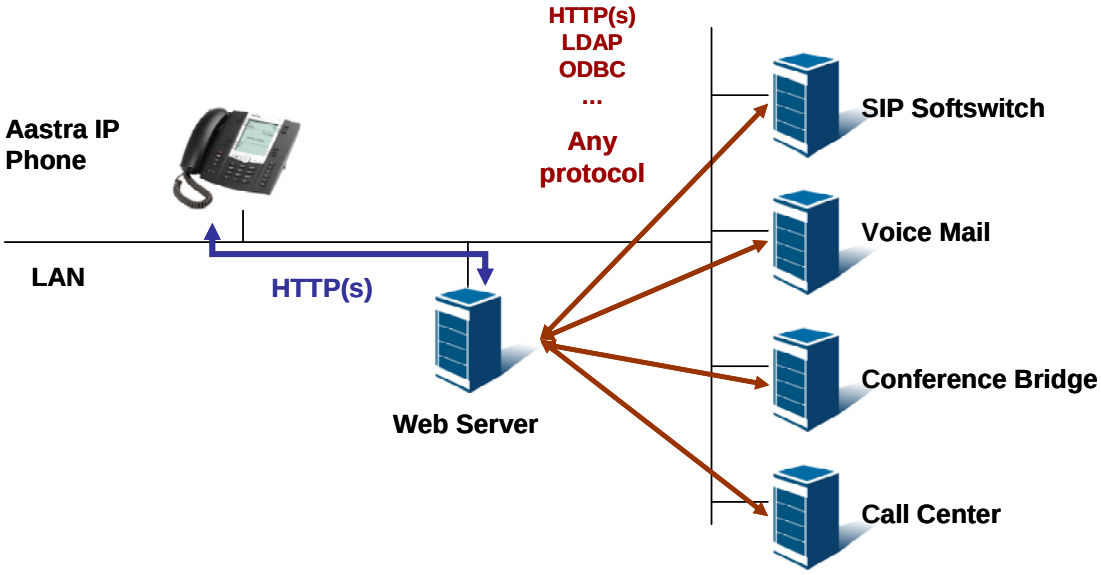


Figure 6: Access to a telephony application

2.5 Development environment

2.5.1 Typical software architecture

The following diagram details the typical architecture of an XML application.

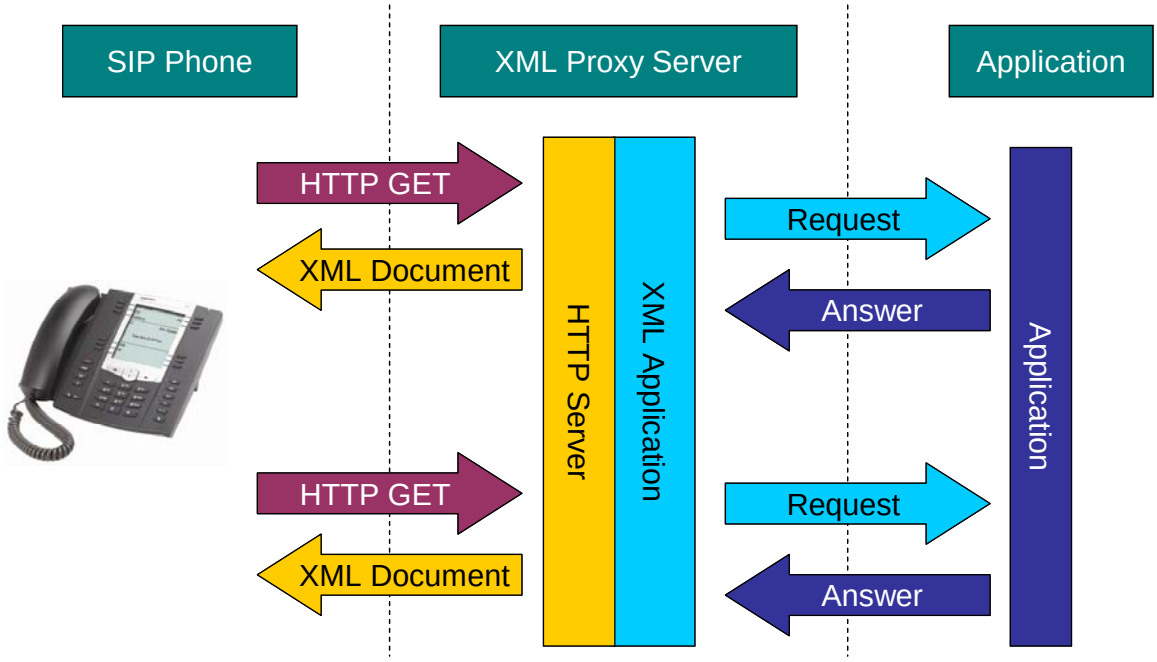


Figure 7: Typical software architecture of an XML application

The XML application is in fact translating requests from the phone to the protocol used by the external application and formats the answer into an XML document that the Aastra SIP phone can interpret.

2.5.2 Web server

There is no constraint in the choice of the Web Server software to be used for Aastra XML applications; the choice is more based on the tools needed for the development such as the script language, the corporate policy or even the cost of the platform.

The most common Web server applications supported are:

- Apache (http://www.apache.org) for Microsoft and Linux Operating systems
- Microsoft IIS for Microsoft Operating systems
- Netscape iPlanet
- …

2.5.3 Scripts/Languages

As for the Web Server, there is no specific constraint on the tools to develop the applications. All the languages supported to develop a Web application are supported to develop XML applications. The most common are:

- Compiled languages: C, C++
- Scripting languages: VBscript, Perl, Python, PHP, asp…

2.5.4 XML Validation tools

A large number of tools are available to validate the XML document you will be sending to the phone, these tools use the XSD schema (provided at chapter 11) to check the syntax of the document.

Example of a web based tool

http://tools.decisionsoft.com/schemaValidate/

2.6 XML format

The text in the Aastra XML objects must be compliant with XML recommendations and special characters must be escape encoded:

Character	Name	Escape Sequence
&	Ampersand	&amp;
“	Quote	&quot;
‘	Apostrophe	&apos;
<	Left angle bracket	&lt;
>	Right angle bracket	&gt;

Figure 8: XML conversion table

2.7 HTTP format

The HTTP message sent to the Aastra SIP Phone must respect HTTP/1.1 and must include the following parameters in the header:

- Content-Length
- Content-Type

Example

```
HTTP/1.1 200 OK                                          } HTTP Header
Date: Tue, 15 May 2007 14:24:33 GMT
Server: Apache/2.0.52 (CentOS)
X-Powered-By: PHP/4.3.11
Content-Length: 564
Connection: close
Content-Type: text/xml

<AastraIPPhoneInputScreen type="string">                 } HTTP body
<Title>Title</Title>
<Prompt>Enter value</Prompt>
<URL>http://myserver.com/script.php</URL>
<Parameter>value</Parameter>
<Default></Default>
</AastraIPPhoneInputScreen>
```

2.8 XML display control and keys

This chapter describes the available part of the display for each Aastra SIP phones as well as the keys that are controlled by the XML objects.

2.8.1 Aastra 53i

The display and keys available for XML applications on an Aastra 53i are:

- 3 lines of 16 characters for the display
- the left and right arrow navigation keys
- the up and down rocker navigation keys

The 3rd line of the display is a command line and will be used to display the labels of the available actions. See chapter 3 for more details on how each XML object will use this line of command.

Depending on the XML object displayed on the phone,

- the left arrow navigation key can also be interpreted as a “cancel” key,
- the right arrow navigation key can also be interpreted as a “next” key,

See chapter 3 for more detailed information on each object.

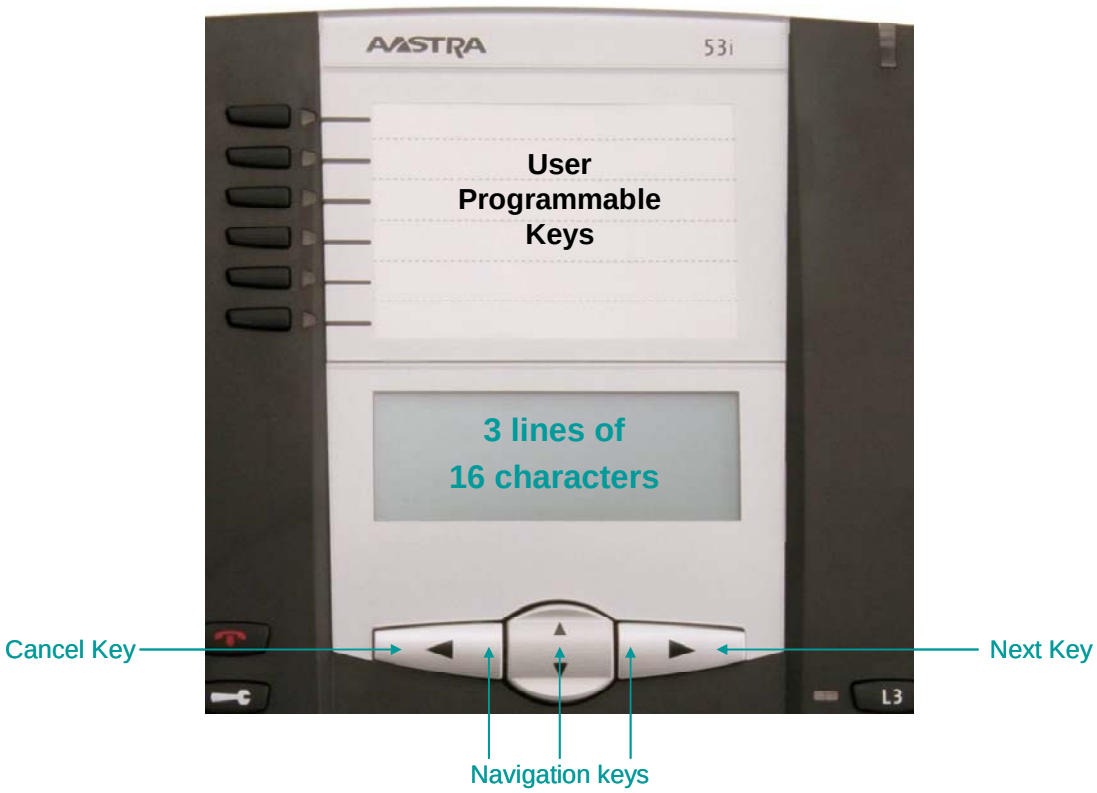


Figure 9: Aastra 53i XML display and keys

2.8.2 Aastra 55i

The display and keys available for XML applications on an Aastra 55i are:

- 5 lines of 22 characters for the display
- the left and right arrow navigation keys
- the up and down rocker navigation keys
- 6 softkeys

Depending on the XML object displayed on the phone,

- the left arrow navigation key can also be interpreted as a “cancel” key,
- the right arrow navigation key can also be interpreted as a “next” key,

See chapter 3 for more detailed information on each object.

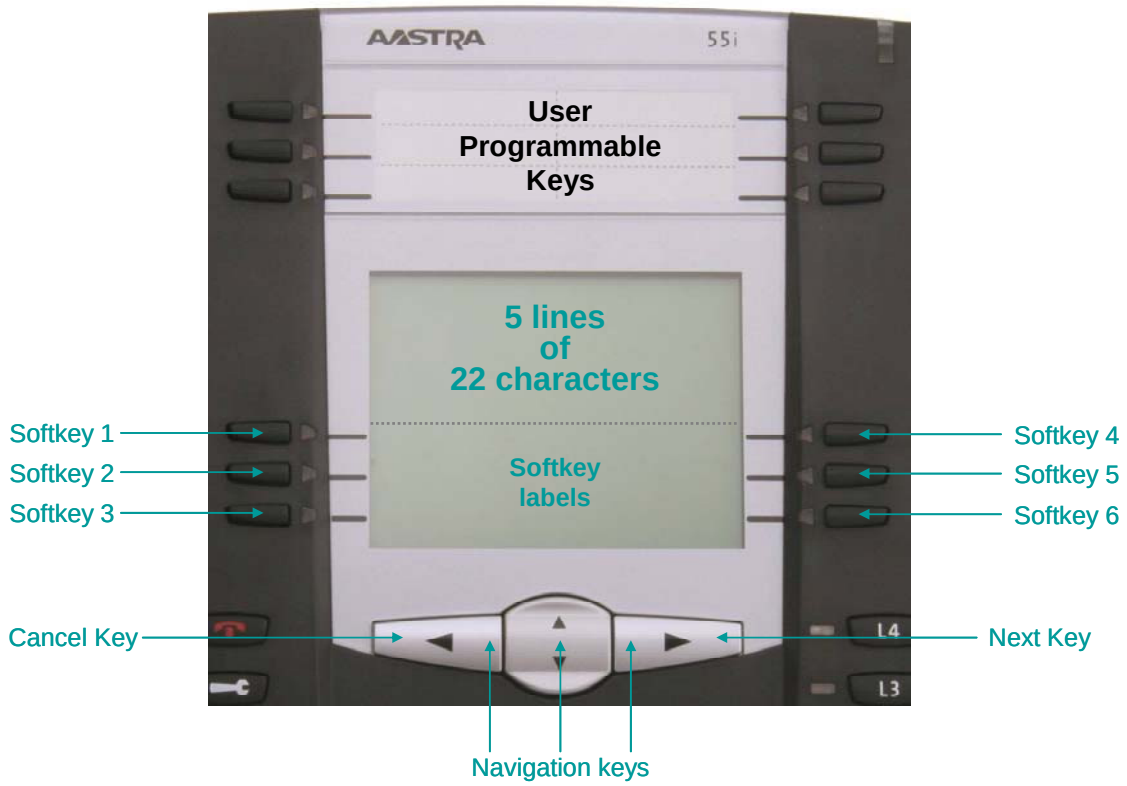


Figure 10: Aastra 55i XML display and keys

2.8.3 Aastra 57i/57iCT

The display and keys available for XML applications on an Aastra 57i/57iCT are:

- 6 lines of 22 characters for the display
- the left and right arrow navigation keys
- the up and down rocker navigation keys
- 6 softkeys

Depending on the XML object displayed on the phone,

- the left arrow navigation key can also be interpreted as a “cancel” key,
- the right arrow navigation key can also be interpreted as a “next” key,

See chapter 3 for more detailed information on each object.

Note: the top 3 lines of the display are not available for XML applications; they are dedicated to the labels of the top softkeys.

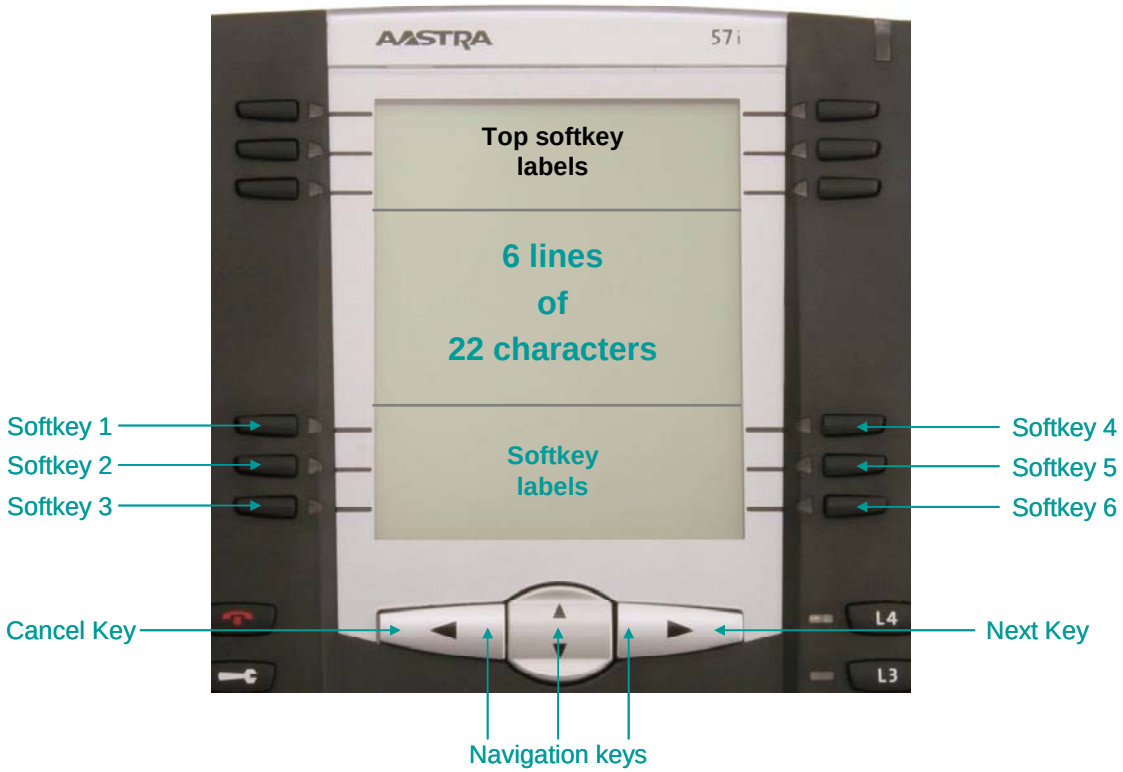


Figure 11: Aastra 57i/57iCT XML display and keys

3 Aastra IP Phone XML Objects

This chapter details all the XML objects supported by Aastra SIP Phones.

Note: the size of an XML object can not exceed 10000 bytes (10 kb).

3.1 TextMenu Object (All models)

The `TextMenu` object allows developers to create a list of menu items on the IP phones. Go-to line support, arrow indicator, and scroll key support are built into these objects, along with the "**Select"** and "**Done"** softkeys for the phones supporting softkeys. The TextMenu object allows users to navigate the application, by linking HTTP requests to menu items.

For more details on the TextMenu behavior using the `Selection` tag or the `Dial` tag, please refer to section 4.6 and 4.7.

Object native interaction

- ***Select*** Executes the content of the URI field assigned to the selected MenuItem;
- ***Exit*** Redisplays the previous XML object present in the phone browser.

53i keys

On the 53i, the object is displayed on one line or one item at a time. The Up and Down arrow keys allow the user to browse the list up and down.

Line selected	Label	Keys	
Title	`Use ^v to view`	Up and Down Arrow	Browse up or down
		Left Arrow	***Exit***
Item	`vNext`	Up and Down Arrow	Browse up or down
	`>Enter`	Right Arrow	***Select***
		Left Arrow	***Exit***

55i/57i/57iCT keys

On the 55i/57i/57iCT, the object is displayed on up to 6 lines, the title stays on the top of the display. The Up and Down arrow keys allow the user to browse the list up and down.

Line selected	Keys	
Item	Up and Down Arrow	Browse up or down
	Right Arrow	***Select***
	Left Arrow	***Exit***

Note:

- the Left Arrow key interaction is disabled if the `LockIn` tag is set to “yes”.
- the Left Arrow key interaction can be modified using the `cancelAction` tag.

55i/57i/57iCT Object Default Softkeys

Position	Label	Interaction	URIs
1	Select	***Select:*** Executes the content of the URI field assigned to the selected MenuItem;	SoftKey:Select
6	Done	***Exit:*** Redisplays the previous XML object present in the phone browser.	SoftKey:Exit

XML Description

```
<AastraIPPhoneTextMenu
    defaultIndex = “some integer”
    destroyOnExit = “yes/no”
    style = “numbered/none/radio”
    Beep = “yes/no”
    Timeout = “some integer”
    LockIn = “yes/no”
    allowAnswer = “yes/no”
    cancelAction = “some URI”
    >
      <Title wrap=”yes/no”>Menu Title</Title>
      <MenuItem base = ”http://base/” icon = “icon index” >
            <Prompt>First Choice</Prompt>
            <URI>http://somepage.xml</URI>
            <Dial>Number to dial<Dial>
            <Selection>Selection</Selection>
      </MenuItem>
      <IconList>
            <Icon index = “int”>Icon:Iconname or HEX string</Icon>
            <!—As many as different icons used in the object -->
      </IconList>
      <!—Additional Menu Items may be added -->
      <!—Additional Softkey Items may be added (55i/57i/57i CT)-->
</AastraIPPhoneTextMenu>
```

XML Document Objects

Document Object	Position	Type	Comments
`AastraIPPhoneTextMenu`	Root tag	Mandatory	Root object
`defaultIndex`	Root tag	Optional	Position of the cursor when the XML object is open. If not specified the arrow is positioned on the first menu item.
`style`	Root tag	Optional	“numbered/none/radio” indicates the style of the TextMenu. Default is “numbered”.

Document Object	Position	Type	Comments
destroyOnExit	Root tag	Optional	"yes/no" indicates if the object is kept or not in the phone browser after exit. If not specified, the object is kept in the browser.
cancelAction	Root tag	Optional	Defines the URI to be called when the user cancels the XML object.
Beep	Root tag	Optional	"yes" or "no" to indicate if a notification beep must be generated by the phone.
Timeout	Root tag	Optional	Overrides the default 45 seconds timeout of the UI XML object. A Timeout set to "0" will disable the timeout feature. See section 4.4.2 for more details
LockIn	Root tag	Optional	If set to "yes", the phone ignores all events that would cause the screen to exit without using the keys defined by the object. Default value is "no". See section 4.4.3 for more details.
allowAnswer	Root tag	Optional	This tag applies only to the non-softkey phones (53i). If set to "yes", the phone will display "Ignore" and "Answer" if the XML object is displayed when the phone is in the ringing state. Default value is "no". See section 6.3 for more details.
Title	Body	Optional	Text to be used as title for the object
Wrap	Title tag	Optional	If set to "yes" the title of the object will be wrapped on 2 lines.
MenuItem	Body	Mandatory	Choice Item (up to 15 instances)
Base	MenuItem tag	Optional	The value of this attribute is pre-pended to the value in the URI tags.
Icon	MenuItem tag	Optional	Index of the icon to be used for this menu entry
Prompt	MenuItem body	Mandatory	Label of the item
URI	MenuItem body	Mandatory	URI to be used if the user press "Select" on this item
Dial	MenuItem body	Optional	Defines what number will be dialed when an offhook action is performed on the phone or if the "Dial2" custom softkey is pressed
Selection	MenuItem body	Optional	This tag must be used in conjunction with custom softkeys. See Section 4.1 for details
IconList	Body	Optional	List of icon definitions

Document Object	Position	Type	Comments
`Icon`	IconList body	Optional	Icon value, it can be “Icon:Iconname” or an hexadecimal string representing the icon. See section 4.1.2 for more details.
`Index`	Icon tag	Optional	Index of the icon must be consistent with the ‘Icon’ used in the MenuItems.
`SoftKey`	Body	Optional	See section 4.1 for details

Limitations on the 53i

- Custom Softkeys are not supported
- Icons are not supported
- Only “numbered” and “none” values are supported for the `Style` tag. The “radio” value will have the same behavior than “none”.

TextMenu styles

Figure 12: “numbered” style TextMenu

Figure 13: “none” style TextMenu

Figure 14: “radio” style TextMenu

XML Example 1

```
<AastraIPPhoneTextMenu>
      <Title>Phone Services</Title>
      <MenuItem base = “http://10.50.10.53/”>
            <Prompt>Traffic Reports</Prompt>
            <URI> rss_to_xml.pl</URI>
```

```
        </MenuItem>
        <MenuItem>
              <Prompt>Employee List</Prompt>
              <URI>employees.xml</URI>
        </MenuItem>
        <MenuItem base =””>
              <Prompt>Weather</Prompt>
              <URI>http://10.50.10.52/weather.pl</URI>
        </MenuItem>
</AastraIPPhoneTextMenu>
```

Resulting Screens (53i)

Figure 15: TextMenu Example 1 (53i)

Resulting Screen (55i/57i/57iCT)

Figure 16: TextMenu Example 1 (55i/57i/57iCT)

XML Example 2

```
<AastraIPPhoneTextMenu>
```

```
      <Title>Phone Services</Title>
      <MenuItem base = "http://10.50.10.53/" icon="1">
            <Prompt>Traffic Reports</Prompt>
            <URI> rss_to_xml.pl</URI>
      </MenuItem>
      <MenuItem icon="2">
            <Prompt>Employee List</Prompt>
            <URI>employees.xml</URI>
      </MenuItem>
      <MenuItem base ="" icon="3">
            <Prompt>Weather</Prompt>
            <URI>http://10.50.10.52/weather.pl</URI>
      </MenuItem>
      <IconList>
            <Icon index="1">045E545E0400</Icon>
            <Icon index="2">FEAA8292FE00</Icon>
            <Icon index="3">C00E60029800</Icon>
      </IconList>
</AastraIPPhoneTextMenu>
```

Resulting Screen

Figure 17: TextMenu Example 2 (55i/57i/57iCT)

3.2 ImageMenu Object (55i/57i/57i CT)

The `ImageMenu` object allows using a bitmap image to serve as a menu. This is desirable when a user wants to display menu choices in some non-ASCII character set or with pictures only. Each menu selection is linked to a keypad key (0-9, *, #).allows developers to create a numerical list of choices.

The image itself is specified as a series of hexadecimal characters. See chapter 4.1.1 for more details on the format of the string.

Note: With firmware 2.1.0 the image is limited to 144x40 pixels.

Default Softkeys

Position	Label	Description	URIs
6	Done	Redisplays the previous XML object present in the phone browser.	SoftKey:Exit

XML Description

```
<AastraIPPhoneImageMenu
    destroyOnExit = “yes/no”
    cancelAction = “some URI”
    Beep = “yes/no”
    Timeout = “some integer”
    LockIn = “yes/no”
    >
      <Image
            verticalAlign = “top,middle,bottom”
            horizontalAlign = “left,middle,right”
            height = “height in pixels”
            width = “width in pixels”
      >Image as hexadecimal characters</Image>
      <!—Base attribute is optional-->
      <URIList base = "http://someserver/">
            <URI key = "0">link1.php</URI>
            <URI key = "#">link3.php</URI>
            <!—Additional URI entries may be added (0-9 and #)-->
      </URIList>
<!—Additional Softkey Items may be added -->
</AastraIPPhoneImageMenu>
```

XML Document Objects

Document Object	Position	Type	Comments
AastraIPPhoneImageMenu	Root tag	Mandatory	Root object
destroyOnExit	Root tag	Optional	“yes/no” indicates if the object is kept or not in the phone browser after exit. If not specified, the object is kept in the browser.
cancelAction	Root tag	Optional	Defines the URI to be called when the user cancels the XML object.
Beep	Root tag	Optional	“yes” or “no” to indicate if a notification beep must be generated by the phone.
Timeout	Root tag	Optional	Overrides the default 45 seconds timeout of the UI XML object. A Timeout set to “0” will disable the timeout feature. See section 4.4.2 for more details
LockIn	Root tag	Optional	If set to “yes”, the phone ignores all events that would cause the screen to exit without using the keys defined by the object. Default value is “no”.

Document Object	Position	Type	Comments
			See section 4.4.3 for more details.
Image	Body	Mandatory	Image to be displayed as a series of hexadecimal characters. See section 4.1.1 for more details.
verticalAlign	Image tag	Optional	Vertical position of the image (“top”, “middle” or “bottom”). If the tag is not specified, the object will use “middle” as a default value.
horizontalAlign	Image tag	Optional	Horizontal position of the image (“left”, “middle” or “right”). If the tag is not specified, the object will use “middle” as a default value.
height	Image tag	Mandatory	Height in pixels. Must match the image height.
width	Image tag	Mandatory	Width in pixels. Must match the image width.
URIList	Body	Mandatory	Master tag of the URI list linked to a keypad key (0-9, * and #)
Base	URIList tag	Optional	The value of this attribute is pre-pended to the value in the URI tags.
URI	URIList body	Mandatory	URI to be used if the user presses the key defined in the “key” tag.
key	URI tag	Mandatory	This tag defines the key that will trigger the selection (0-9 and #).
SoftKey	Body	Optional	See section 4.1 for details

XML Example

```
<AastraIPPhoneImageMenu destroyOnExit="yes">
<Image height="40"
width="144">ffffffc02ffffffee4ffffbfffc05fffe7ff7a7fffffffeffeeb
d7fffffea6bcffffe796f3feff6fa289f0a86f4866fa20df42414595dd0134f803
7ed1637f0e2522b2dd003b6eb936f05fffbd4f4107bba6eb0080e93715000010b75
4001281271408c640252081b1b22500013c5c66201368004e04467520dc11067152
b82094d418e10024720580549478010500260153002013140002 0a05c91088b002b
08c21c0000c200000001fe80000000000000001c0000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000200410000040083000 00ff08500000
000c900000000710000000000000014014000001401400000140140000014014000
0001401400000000000000007c0ff00000c3088000008108800000810880000 0c307
0000006200000000000 3f000001e020000003302000000210000000033 01e000001
e033000000002100000 3f033000002001e000002000000000000 1e000c03fc33003
c013021007c02101201f00330ff03f001e000039000003e039001e00103f0033001
01f8021003007c03303f003c01e000000c00001e001c03f033007802002100f0020
02103e000001203c401702003cc0290ff039c02902101fc02b000007c03f01a003c
020039018c0ff02d03c402102703c400001203ec01e000026402b00002640290000
26c029000027c01a000033800000003380000000310000000030000000030003f0
0003fc03000003fc02000003fc020000030001f000030000000030001e00003000
```

```
2b000030002900003fc02900003fc01a00003f000000003100300000031c01e00003
1f003000033f81e00003f383000001e081e000008c003000003c01e00000fc03000
001f000000003d001a000039003900003900 2d00003f002700000f8012000007c00
0000001c000000000400000000000000000000000000000</Image>
<URIList>
<URI key="1">http://myserver.com?choice=1</URI>
<URI key="2">http://myserver.com?Choice=2</URI>
</URIList>
<SoftKey index="1" icon="1">
<Label>Label</Label>
<URI>http://myserver.com/script.php?action=1</URI>
</SoftKey>
<SoftKey index="6">
<Label>Exit</Label>
<URI>SoftKey:Exit</URI>
</SoftKey>
<IconList>
<Icon index="1">Icon:Envelope</Icon>
</IconList>
</AastraIPPhoneImageMenu>
```

Resulting Screen

Figure 18: ImageMenu Example

3.3 TextScreen Object (All models)

The `TextScreen` object can be used to display text. The screen word-wraps appropriately and can scroll to display a message longer then the physical display.

Object native interaction

- ***Done*** Redisplays the previous XML object present in the phone browser.

53i keys

On the 53i, the object is displayed on 2 lines. The Up and Down arrow keys allow the user to browse the rest of the text.

Line selected	Label	Keys	
Title/Text	vNext	Up and Down Arrow	Browse up or down
	>Done	Right Arrow	***Done***
		Left Arrow	***Exit***

55i/57i/57iCT keys

On the 55i/57i/57iCT, the object is displayed on up to 6 lines, the title stays on the top of the display. The Up and Down arrow keys allow the user to browse the rest of the text.

Line selected	Keys	
Text	Up and Down Arrow	Browse up or down
	Right Arrow	***Done***
	Left Arrow	***Exit***

Note:

- the Left Arrow key interaction is disabled if the `LockIn` tag is set to "yes".
- the Left Arrow key interaction can be modified using the `cancelAction` tag.
- the right arrow (Done) interaction can be modified using the `doneAction` tag.

55i/57i/57iCT Default Softkey(s)

Position	Label	Description	URIs
6	Done	Redisplays the previous XML object present in the phone browser.	SoftKey:Exit

XML Description

```
<AastraIPPhoneTextScreen
    destroyOnExit = "yes/no"
    cancelAction = "some URI"
    doneAction = "some URI"
    Beep = "yes/no"
    Timeout = "some integer"
    allowAnswer = "yes/no"
    LockIn = "yes/no"
    >
      <Title wrap="yes/no">Screen Title</Title>
      <Text>The screen text goes here</Text>
<!—Additional Softkey Items may be added (55i/57i/57i CT)-->
</AastraIPPhoneTextScreen>
```

XML Document Objects

Document Object	Position	Type	Comments
`AastraIPPhoneTextScreen`	Root tag	Mandatory	Root object

Document Object	Position	Type	Comments
destroyOnExit	Root tag	Optional	"yes/no" indicates if the object is kept or not in the phone browser after exit. If not specified, the object is kept in the browser.
cancelAction	Root tag	Optional	Defines the URI to be called when the user cancels the XML object.
doneAction	Root tag	Optional	Defines the URI to be called when the user selects the "Done" softkey.
Beep	Root tag	Optional	"yes" or "no" to indicate if a notification beep must be generated by the phone.
Timeout	Root tag	Optional	Overrides the default 45 seconds timeout of the UI XML object. A Timeout set to "0" will disable the timeout feature. See section 4.4.2 for more details
LockIn	Root tag	Optional	If set to "yes", the phone ignores all events that would cause the screen to exit without using the keys defined by the object. Default value is "no". See section 4.4.3 for more details.
allowAnswer	Root tag	Optional	This tag applies only to the non-softkey phones (53i). If set to "yes", the phone will display "Ignore" and "Answer" if the XML object is displayed when the phone is in the ringing state. Default value is "no". See section 6.3 for more details.
Title	Body	Optional	Label to be used as title for the object
Wrap	Title tag	Optional	If set to "yes" the title of the object will be wrapped on 2 lines.
Text	Body	Mandatory	Text to be displayed.
SoftKey	Body	Optional	See section 3.9 for details

Limitations on the 53i

- Custom Softkeys are not supported

XML Example

```
<AastraIPPhoneTextScreen>
      <Title>Screen Object</Title>
      <Text>The screen object can be implemented similar to the
firmware info screen. Note that white space is preserved in XML so
```

```
the display should word-wrap appropriately. Only three lines can
display at a time.</Text>
</AastraIPPhoneTextScreen>
```

Resulting Screens (53i)

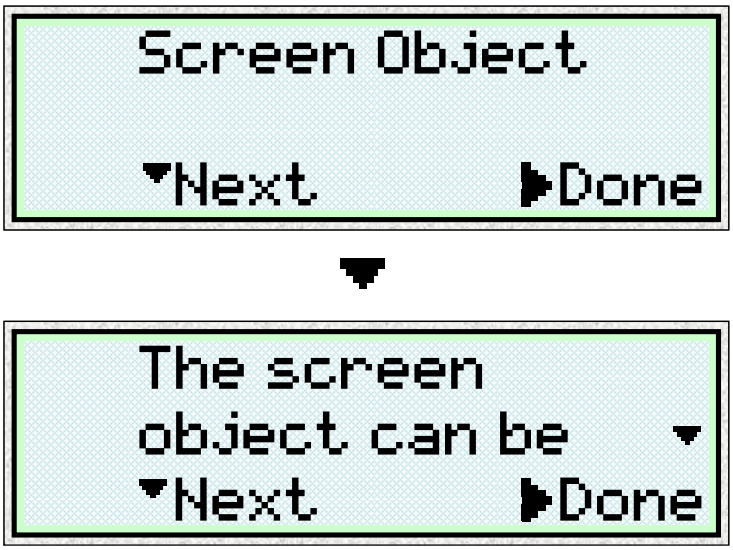


Figure 19: TextScreen Example (53i)

Resulting Screen (55i/57i/57iCT)

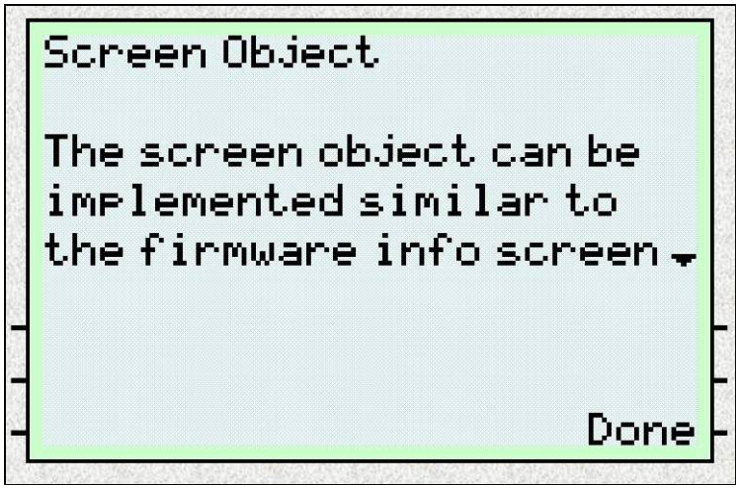


Figure 20: TextScreen Example (55i/57i/57iCT)

3.4 FormattedTextScreen Object (All Models)

The `FormattedTextScreen` object allows the XML designer to display formatted (alignment, size and scrolling) text.

This text is divided into 3 distinct blocks, any of which can be empty.

- The first block is displayed at the top of the display and contains static text. This block takes up as many lines as the XML object specifies and can range from 0 up to the size of the physical screen.

- The next block, displayed below the first block, displays scrolling text and takes up as many lines as the designer specifies up to the size of the screen.

- The final block of contains static text and will take up whatever lines are left on the screen.

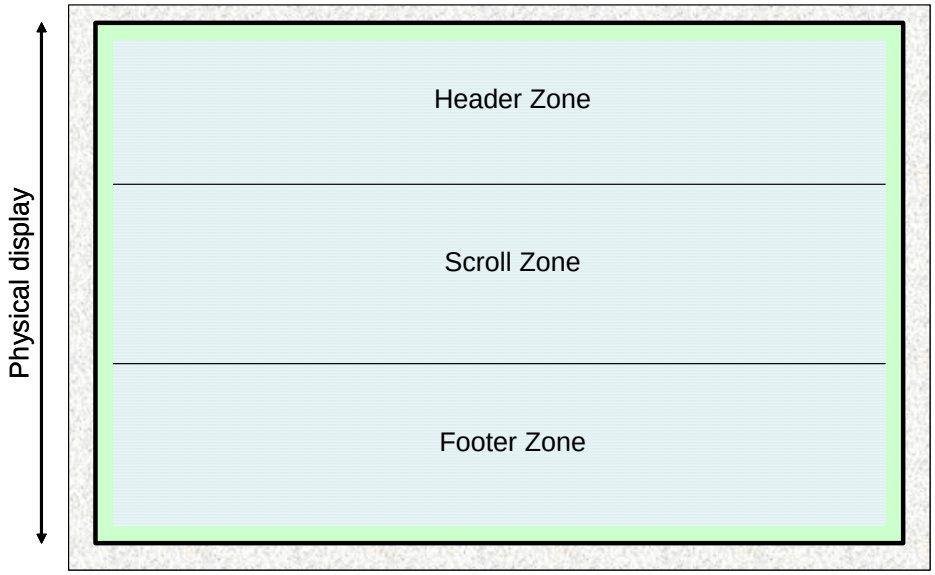


Figure 21: FormattedTextScreen layout

Object native interaction

- ***Done*** Redisplays the previous XML object present in the phone browser.

53i keys

On the 53i, the object is displayed on 2 lines. The Up and Down arrow keys allow the user to browse the rest of the text.

Line selected	Label	Keys	
Text		Up and Down Arrow	Browse up or down (if scrollable)
	>Done	Right Arrow	***Done***
		Left Arrow	***Exit***

55i/57i/57iCT keys

On the 55i/57i/57iCT, the object is displayed on up to 6 lines, the title stays on the top of the display. The Up and Down arrow keys allow the user to browse the rest of the text.

Line selected	Keys	
Text	Up and Down Arrow	Browse up or down (if scrollable)
	Right Arrow	***Done***

	Left Arrow	***Exit***

Notes:

- the Left Arrow key interaction is disabled if the `LockIn` tag is set to “yes”.
- the Left Arrow key interaction can be modified using the `cancelAction` tag.
- the right arrow (Done) interaction can be modified using the `doneAction` tag.

(55i/57i/57iCT) Default Softkeys

Position	Label	Description	URIs
6	Done	Redisplays the previous XML object present in the phone browser.	SoftKey:Exit

XML Description

```
<AastraIPPhoneFormattedTextScreen
    destroyOnExit = “yes/no”
    cancelAction = “some URI”
    doneAction = “some URI”
    Beep = “yes/no”
    Timeout = “some integer”
    allowAnswer = “yes/no”
    LockIn = “yes/no”
>
     <Line  Size=”normal/double”
            Align=”left/center/right”
      >A line of static text</Line>
      <!—Additional Lines may be added -->
     <Scroll Height=”2”>
            <Line Size=”normal/double”
                  Align=”left/center/right”
            >Scrolling text</Line>
            <!—Additional Lines may be added -->
    </Scroll>
    <Line>Some static footer text</Line>
    <!—Additional Lines may be added -->
<!—Additional Softkey Items may be added -->
</AastraIPPhoneFormattedTextScreen>
```

Notes:

- Any lines that would display past the bottom of the screen will be ignored.
- Double height text will not display with the top or bottom cut off.
- Text that extends past the edge of the screen will be cropped to the last fully displayed word.

XML Document Objects

Document Object	Position	Type	Comments
`AastraIPPhoneTextScreen`	Root tag	Mandatory	Root object
`destroyOnExit`	Root tag	Optional	"yes/no" indicates if the object is kept or not in the phone browser after exit. If not specified, the object is kept in the browser.
`cancelAction`	Root tag	Optional	Defines the URI to be called when the user cancels the XML object.
`doneAction`	Root tag	Optional	Defines the URI to be called when the user selects the "Done" softkey.
`Beep`	Root tag	Optional	"yes" or "no" to indicate if a notification beep must be generated by the phone.
`Timeout`	Root tag	Optional	Overrides the default 45 seconds timeout of the UI XML object. A Timeout set to "0" will disable the timeout feature. See section 4.4.2 for more details
`LockIn`	Root tag	Optional	If set to "yes", the phone ignores all events that would cause the screen to exit without using the keys defined by the object. Default value is "no". See section 4.4.3 for more details.
`allowAnswer`	Root tag	Optional	This tag applies only to the non-softkey phones (53i). If set to "yes", the phone will display "Ignore" and "Answer" if the XML object is displayed when the phone is in the ringing state. Default value is "no". See section 6.3 for more details.
`Line`	Body	Optional	Text to be displayed on the line. If the text is larger than the display, line is cropped to the last word.
`Size`	Line tag	Optional	Size of the font for the line, "normal" for the regular font, "double" for a double height font. "normal" is the default value if not

Document Object	Position	Type	Comments
			specified.
Align	Line tag	Optional	Alignment of the line, “left”, “right” or “center”. If not specified the default value is “left”.
Scroll	Body	Optional	Defines the scrolling section of the display.
Height	Scroll tag	Optional	Specifies the height of the scroll zone. If not specified, the phone uses the remaining lines of the physical screen and does not allow footer lines.
Line	Scroll Body	Optional	Text to be displayed on the line in the scrolled zone. If the text is larger than the display, line is cropped to the last word.
Size	Line tag	Optional	Size of the font for the scrolled line, “normal” for the regular font, “double” for a double height font. “normal” is the default value if not specified.
Align	Line tag	Optional	Alignment of the scrolled line, “left”, “right” or “center”. If not specified the default value is “left”.
SoftKey	Body	Optional	See section 3.9 for details

Limitations on the 53i

- Custom Softkeys are not supported
- Only 2 lines are available to display the object.
    - A double size line on line 1 does not allow any scrolling
    - The scroll height can not be more than 2.

XML Example (53i)

```
<AastraIPPhoneFormattedTextScreen destroyOnExit = “yes”>
     <Scroll Height=”2”>
            <Line>Line 1</Line>
            <Line>Line 2</Line>
            <Line Size=”double”>Line 3</Line>
            <Line>Line 4</Line>
            <Line>Line 5</Line>
    </Scroll>
    <Line Align=”center”>Footer</Line>
</AastraIPPhoneFormattedTextScreen>
```

Resulting Screen (53i)

Figure 22: FormattedTextScreen Example (53i)

XML Example (55i/57i/57iCT)

```
<AastraIPPhoneFormattedTextScreen destroyOnExit = "yes">
     <Line Size="double" Align="center">Formatted Screen</Line>
     <Scroll Height="2">
            <Line>Scrolling text1</Line>
            <Line>Scrolling text2</Line>
            <Line>Scrolling text3</Line>
            <Line>Scrolling text4</Line>
            <Line>Scrolling text5</Line>
    </Scroll>
    <Line Align="center">Footer</Line>
</AastraIPPhoneFormattedTextScreen>
```

Resulting Screen (55i/57i/57iCT)

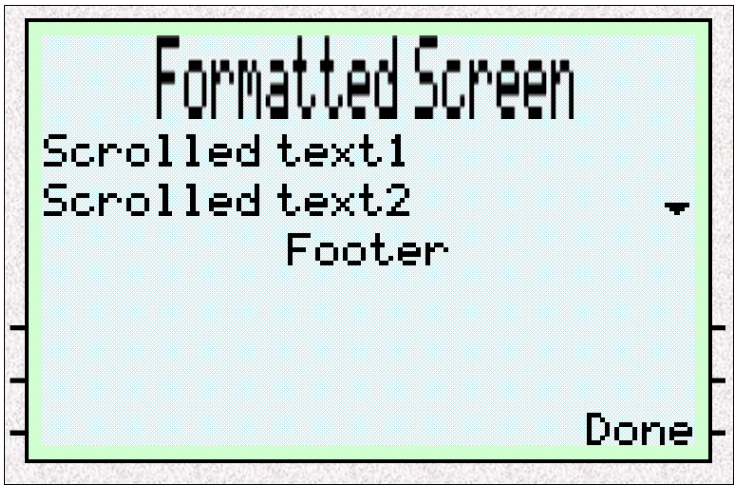


Figure 23: FormattedTextScreen Example (55i/57i/57iCT)

3.5 ImageScreen Object (55i/57i/57i CT)

The `ImageScreen` object can be used to display single image bitmap. The user can specify where the image should be placed by setting horizontal and vertical alignment of the upper left hand corner, along with the height and width of the image.

The image itself is specified as a series of hexadecimal characters. See chapter 4.1.1 for more details.

Note: With firmware 2.0 the image size is limited to 144x40 pixels.

Default Softkeys

Position	Label	Description	URIs
6	Done	Redisplays the previous XML object present in the phone browser.	SoftKey:Exit

XML Description

```
<AastraIPPhoneImageScreen
    destroyOnExit = “yes/no”
    cancelAction = “some URI”
    Beep = “yes/no”
    Timeout = “some integer”
    LockIn = “yes/no”
    >
      <Image
            verticalAlign = “top,middle,bottom”
            horizontalAlign = “left,middle,right”
            height = “height in pixels”
            width = “width in pixels”
      >Image as hexadecimal characters</Image>
<!—Additional Softkey Items may be added -->
</AastraIPPhoneImageScreen>
```

XML Document Objects

Document Object	Position	Type	Comments
`AastraIPPhoneImageScreen`	Root tag	Mandatory	Root object
`destroyOnExit`	Root tag	Optional	“yes/no” indicates if the object is kept or not in the phone browser after exit. If not specified, the object is kept in the browser.
`cancelAction`	Root tag	Optional	Defines the URI to be called when the user cancels the XML object.
`Beep`	Root tag	Optional	“yes” or “no” to indicate if a notification beep must be generated by the phone.
`Timeout`	Root tag	Optional	Overrides the default 45 seconds timeout of the UI XML object. A Timeout set to “0” will disable the timeout feature. See section 4.4.2 for more details
`LockIn`	Root tag	Optional	If set to “yes”, the phone ignores all events that would cause the screen to exit without using the keys defined by the object. Default value is “no”. See section 4.4.3 for more details.
`Image`	Body	Mandatory	Image to be displayed as a series of hexadecimal characters. See section 4.1.1 for more details.
`verticalAlign`	Image tag	Optional	Vertical position of the image (“top”, “middle” or “bottom”). If the tag is not specified, the object will use “middle” as a default value.
`horizontalAlign`	Image tag	Optional	Horizontal position of the image (“left”, “middle” or “right”). If the tag is not specified, the object will use “middle” as a default value.
`Height`	Image tag	Mandatory	Height in pixels. Must match the image height.
`Width`	Image tag	Mandatory	Width in pixels. Must match the image width.
`SoftKey`	Body	Optional	See section 3.9 for details

XML Example

```
<AastraIPPhoneImageScreen destroyOnExit="yes">
<Image height="40"
width="40">ffffffc02ffffffee4ffffbfffc05fffe7ff7a7fffffffeffeebd
7fffffea6bcffffe796f3feff6fa289f0a86f4866fa20df42414595dd0134f8037
ed1637f0e2522b2dd003b6eb936f05fffbd4f4107bba6eb0080e93715000010b754
```

```
001281271408c640252081b1b22500013c5c66201368004e04467520dc11067152b
82094d418e10024720580549478010500260153002093140002 0ac5c91088b0f2b0
8c21c07d0c2006009fdfe81f80efe0107fe0fb1c3ffff8ffc3fffef8f7febffbfcf
87ffbff64</Image>
<SoftKey index="1" icon="1">
<Label>Mail</Label>
<URI>http://myserver.com/script.php?action=1</URI>
</SoftKey>
<SoftKey index="6">
<Label>Exit</Label>
<URI>SoftKey:Exit</URI>
</SoftKey>
<IconList>
<Icon index="1">Icon:Envelope</Icon>
</IconList>
</AastraIPPhoneImageScreen>
```

Resulting Screen

Figure 24: ImageScreen Example

3.6 InputScreen Object – Single Input field (All models)

The InputScreen object allows developers to create a screen capable of gathering user input. The Aastra IP phones support seven input types:

- IP Addresses,
- Numbers (integers plus * and #),
- Strings,
- Dates (US and international format)
- Time (US and international format)

Each parameter has a URL tag that is used to send information back to the HTTP server. The label in the parameter tag is appended to the address in the URL tag and sent via HTTP GET.

Object native interactions

- ***Done*** Executes the content of the URI field assigned to the selected MenuItem;
- ***Cancel*** Completes the user input by submitting the programmed URI and value.

53i keys

On the 53i, the object is displayed on two lines, one line for the prompt message and one line for the input field, the title is not displayed.

Line selected	Label	Keys	
Input field	^Cancel	Up Arrow	***Exit***
	vDone	Down Arrow	***Done***
		Right Arrow	Next Character
		Left Arrow	Previous Character
		prgkey2 (Delete)	Backspace

Note:

- the Up Arrow key interaction is disabled if the `LockIn` tag is set to "yes".
- the Up Arrow key interaction can be modified using the `cancelAction` tag.

55i/57i/57iCT keys

Line selected	Keys	
Input field	Right Arrow	Next Character
	Left Arrow	Previous Character

55i/57i/57iCT Common default Softkeys

Position	Label	Description	URIs
5	Done	Completes the user input by submitting the programmed URI and value.	SoftKey:Submit
6	Cancel	Redisplays the previous XML object present in the phone browser. Does not appear if `LockIn` set to "yes".	SoftKey:Exit

XML Description

```
<AastraIPPhoneInputScreen
    type = "IP/string/number/timeUS/timeInt/dateUS/dateInt"
    password = "yes/no"
    editable = "yes/no"
    destroyOnExit = "yes/no"
    cancelAction = "some URI"
    Beep = "yes/no"
    Timeout = "some integer"
    allowAnswer = "yes/no"
    LockIn = "yes/no"
>
      <Title wrap="yes/no">Title string</Title>
      <Prompt>Guidance for the input</Prompt>
```

```
      <URL>Target receiving the input</URL>
      <Parameter>name of the parameter added to URL</Parameter>
      <Default>Default Value</Default>
<!—Additional Softkey Items may be added -->
</AastraIPPhoneInputScreen>
```

XML Document Objects

Document Object	Position	Type	Comments
AastraIPPhoneInputScreen	Root tag	Mandatory	Root object
Type	Root tag	Mandatory	Specifies the type of input, possible values are “IP”, “string”, “number”, “timeUS”, “timeInt”, “dateUS”, “dateInt”.
Password	Root tag	Optional	Specifies if the input is masked by “*” characters. Default value is “no”
Editable	Root tag	Optional	Specifies if the user is allowed to modify the input. Default value is “yes”
destroyOnExit	Root tag	Optional	“yes/no” indicates if the object is kept or not in the phone browser after exit. If not specified, the object is kept in the browser.
cancelAction	Root tag	Optional	Defines the URI to be called when the user cancels the XML object.
Beep	Root tag	Optional	“yes” or “no” to indicate if a notification beep must be generated by the phone.
Timeout	Root tag	Optional	Overrides the default 45 seconds timeout of the UI XML object. A Timeout set to “0” will disable the timeout feature. See section 4.4.2 for more details
LockIn	Root tag	Optional	If set to “yes”, the phone ignores all events that would cause the screen to exit without using the keys defined by the object. Default value is “no”. See section 4.4.3 for more details.
allowAnswer	Root tag	Optional	This tag applies only to the non-softkey phones (53i). If set to “yes”, the phone will display “Ignore” and “Answer” if the XML object is displayed when the phone is in the ringing state. Default value is “no”. See section 6.3 for more details.

Document Object	Position	Type	Comments
`Title`	Body	Optional	Text to be used as title for the object
`Wrap`	Title tag	Optional	If set to “yes” the title of the object will be wrapped on 2 lines.
`Prompt`	Body	Mandatory	Text to be displayed as guidance for the user input.
`URL`	Body	Mandatory	URI called when user completes his input.
`Parameter`	Body	Mandatory	Name of the parameter to be added to the URL after input is complete.
`Default`	Body	Mandatory	Default value to be displayed in the input field.
`SoftKey`	Body	Optional	See section 4.1 for more details

Limitations on the 53i

- Custom Softkeys are not supported
- `Title` tag is not displayed

3.6.1 Input Type: IP

When the type is set to IP, the user input is restricted to integers only. The phone will validate the user input; if an invalid IP address is entered, nothing will be sent to the server and the user will receive an error message.

55i/57i/57iCT Default Softkeys

Position	Label	Description	URIs
1	Backspace	Deletes the character before the cursor in the input field.	SoftKey:BackSpace
2	Dot “.”	Inserts a “.” in the user input at the cursor position	SoftKey:Dot
5	Done	Completes the user input by submitting the programmed URI and value.	SoftKey:Submit
6	Cancel	Redisplays the previous XML object present in the phone browser. Does not appear if `LockIn` set to “yes”.	SoftKey:Exit

XML Example

```
<AastraIPPhoneInputScreen type = ”IP”>
      <Title>Proxy Server</Title>
      <Prompt>Server IP:</Prompt>
      <URL>http://10.50.10.53/script.pl</URL>
```

```
      <Parameter>proxy</Parameter>
      <Default></Default>
<AastraIPPhoneInputScreen>
```

Note: In this example, when the user press "Done" on the phone after entering "192.168.0.100", the phone will call the following URL

"http://10.50.10.53/script.pl?proxy=192.168.0.100".

Resulting Screen (53i)

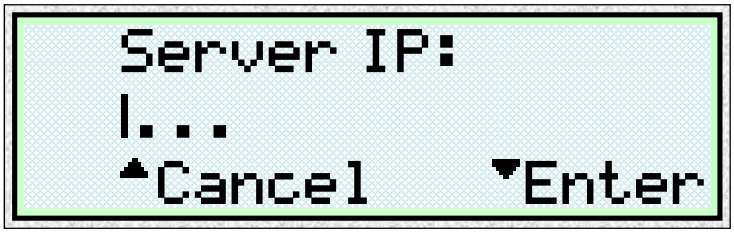


Figure 25: InputScreen "IP" Example (53i)

Resulting Screen (55i/57i/57iCT)

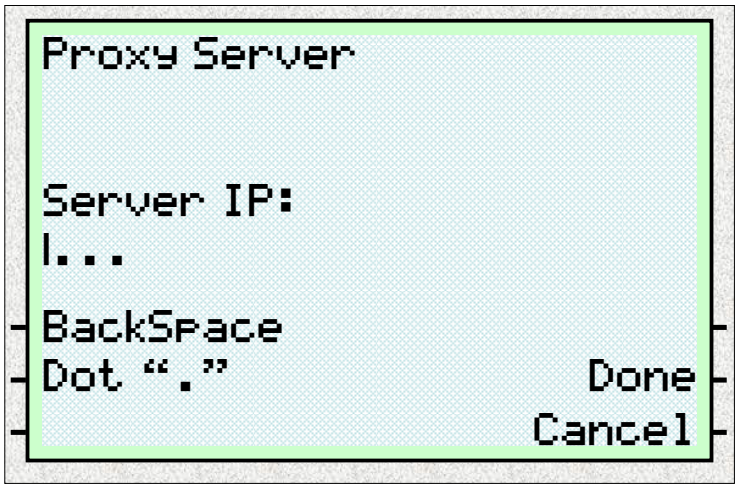


Figure 26: InputScreen "IP" Example (55i/57i/57iCT)

3.6.2 Input Type: Number

Like an IP screen, a number input screen restricts the user to numbers only. However, no type of validation is performed on the user input.

55i/57i/57iCT Default Softkeys

Position	Label	Description	URIs
1	Backspace	Deletes the character before the cursor in the input field.	SoftKey:BackSpace
5	Done	Completes the user input by submitting the programmed URI and value.	SoftKey:Submit

6	Cancel	Redisplays the previous XML object present in the phone browser. Does not appear if `LockIn` set to “yes”.	SoftKey:Exit

XML Example

```
<AastraIPPhoneInputScreen type = ”number”>
      <Title>Proxy Port</Title>
      <Prompt>Port:</Prompt>
      <URL>http://10.50.10.53/script.pl</URL>
      <Parameter>port</Parameter>
      <Default>5060</Default>
<AastraIPPhoneInputScreen>
```

Resulting Screen (53i)

Figure 27: InputScreen “Number” Example (53i)

Resulting Screen (55i/57i/57iCT)

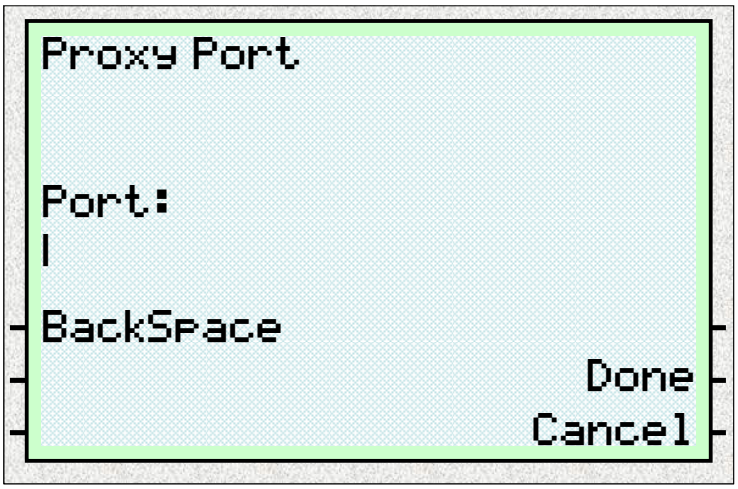


Figure 28: InputScreen “Number” Example (55i/57i/57iCT)

Note: In this example, when the user presses “Done” on the phone after entering “5060”, the phone will call the following URL “`http://10.50.10.53/script.pl?port=5060`”.

3.6.3 Input Type: String

When the input type is set to string, the user can enter uppercase letters, lowercase letters, symbols, and numbers.

53i input

- The user can scroll through some available input symbols

- o . : ; , = _ , – ‘ & ( ) ” by pressing the 1 key repeatedly.
- o # / \ @ by pressing the # key repeatedly.
- o * Space by pressing the * key repeatedly

- Keys 2-9 scroll through the keypad letters in upper case and lower case (e.g. pressing repeatedly 2 will scroll through ABC2abc or abc2ABC)

55i/57i/57iCT input

- The input mode can be switched via the Mode Key (Upper Case, Lower Case and Digits).
- The user can scroll through some available input symbols
  - o . : ; , = _ , – ‘ & ( ) ” by pressing the 1 key repeatedly.
  - o # / \ @ by pressing the # key repeatedly.
  - o * Space by pressing the * key repeatedly
- In Upper Case or Lower Case mode, keys 2-9 scroll through the keypad letters (e.g. pressing repeatedly 2 in Upper case mode will scroll through ABC2)

55i/57i/57iCT Default Softkeys

Position	Label	Description	URIs
1	Backspace	Deletes the character before the cursor in the input field.	SoftKey:BackSpace
2	Dot “.”	Inserts a “.” in the user input at the cursor position	SoftKey:Dot
3	ABC>	Toggle between input modes, “ABC”, “123”, “abc”.	SoftKey:ChangeMode
4	NextSpace	Inserts a space in the user input at the cursor position	SoftKey:NextSpace
5	Done	Completes the user input by submitting the programmed URI and value.	SoftKey:Submit
6	Cancel	Redisplays the previous XML object present in the phone browser. Does not appear if `LockIn` set to “yes”.	SoftKey:Exit

XML Example

```
<AastraIPPhoneInputScreen
  type = ”string”
  password = “yes”>
      <Title>SIP Settings</Title>
      <Prompt>Enter something</Prompt>
      <URL>http://10.50.10.53/script.pl</URL>
      <Parameter>passwd</Parameter>
      <Default></Default>
<AastraIPPhoneInputScreen>
```

Resulting Screen (53i)

Figure 29: InputScreen “String” Example (53i)

Resulting Screen (55i/57i/57iCT)

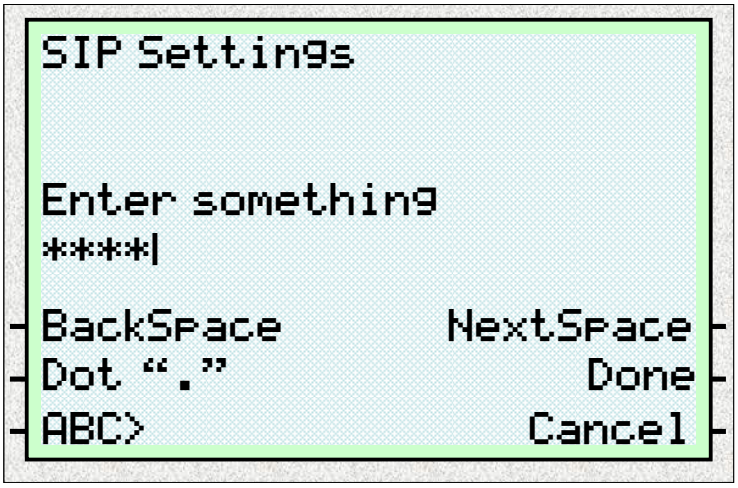


Figure 30: InputScreen “String” Example (55i/57i/57iCT)

Note: In this example, when the user press “Done” on the phone after entering “12345”, the phone will call the following URL “http://10.50.10.53/script.pl?passwd=12345”.

3.6.4 Input Type: timeUS

When the input type is set to “`timeUS`”, the user can enter a time using the US format with 12 hours cycle (HH:MM:SS AM/PM with HH from 00 to 12).

The user navigates between the various fields using the left and right navigation arrow keys, the toggle between AM and PM is done using the right navigation arrow key with the cursor positioned right before the AM/PM field.

Notes:

- the “`password`” attribute has no effect on this input type.
- the format of the “`Default`” attribute must be HH:MM:SS XX where XX is AM or PM and HH between 00 and 12. If the “`Default`” tag is empty, the phone displays “12:00:00 AM”.

55i/57i/57iCT Default Softkeys

Position	Label	Description	URIs
5	Done	Completes the user input by submitting	SoftKey:Submit

		the programmed URI and value.	
6	Cancel	Redisplays the previous XML object present in the phone browser. Does not appear if `LockIn` set to “yes”.	SoftKey:Exit

XML Example

```
<AastraIPPhoneInputScreen type = ”timeUS”>
      <Title>Time US</Title>
      <Prompt>Enter time</Prompt>
      <URL>http://10.50.10.53/script.pl</URL>
      <Parameter>time</Parameter>
      <Default></Default>
<AastraIPPhoneInputScreen>
```

Resulting Screen (53i)

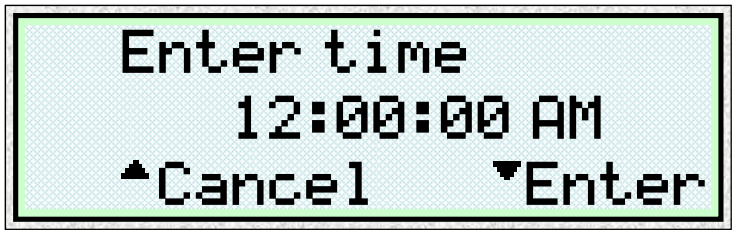


Figure 31: InputScreen “TimeUS” Example (53i)

Resulting Screen (55i/57i/57iCT)

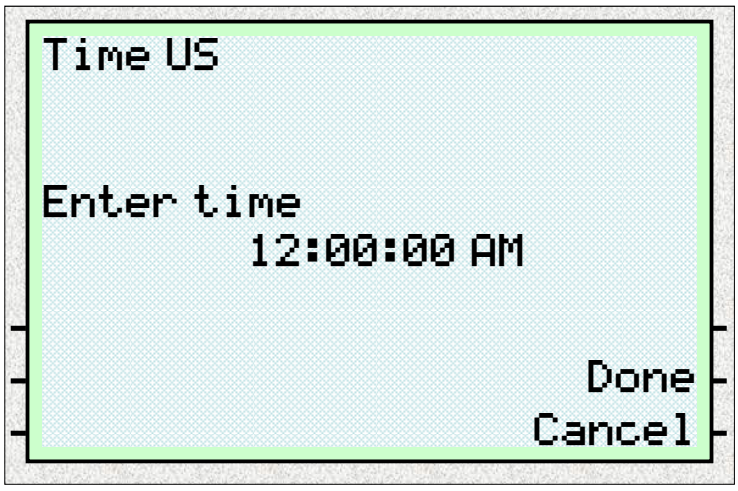


Figure 32: InputScreen “TimeUS” Example (55i/57i/57iCT)

Note: In this example, when the user press “Done” on the phone after entering “10:20:00 AM”, the phone will call the following URL “http://10.50.10.53/script.pl?time=10:20:00AM”.

3.6.5 Input Type: timeInt

When the input type is set to "`timeInt`", the user can enter a time using the international format with a 24 hours cycle (HH:MM:SS HH from 00 to 23).

The user navigates between the various fields using the left and right navigation arrow keys.

Notes:

- the "`password`" attribute has no effect on this input type.
- the format of the "`Default`" attribute must be HH:MM:SS with HH from 00 to 23. If the "`Default`" tag is empty, the phone displays "00:00:00".

55i/57i/57iCT Default Softkeys

Position	Label	Description	URIs
5	Done	Completes the user input by submitting the programmed URI and value.	SoftKey:Submit
6	Cancel	Redisplays the previous XML object present in the phone browser. Does not appear if `LockIn` set to "yes".	SoftKey:Exit

XML Example

```
<AastraIPPhoneInputScreen type = "timeInt">
      <Title>Time International</Title>
      <Prompt>Enter time</Prompt>
      <URL>http://10.50.10.53/script.pl</URL>
      <Parameter>time</Parameter>
      <Default></Default>
<AastraIPPhoneInputScreen>
```

Resulting Screen (53i)

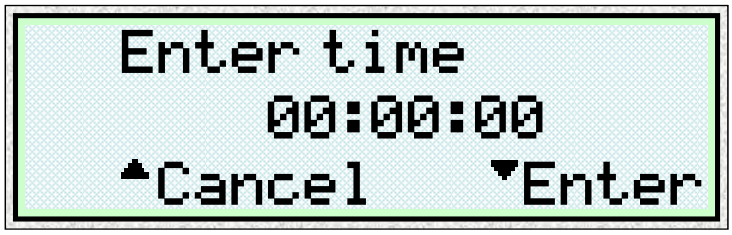


Figure 33: InputScreen "TimeInt" Example (53i)

Resulting Screen (55i/57i/57iCT)

Figure 34: InputScreen "TimeInt" Example (55i/57i/57iCT)

Note: In this example, when the user press "Done" on the phone after entering "14:20:00", the phone will call the following URL "http://10.50.10.53/script.pl?time=14:20:00".

3.6.6 Input Type: dateUS

When the input type is set to "`dateUS`", the user can enter a date using the US format (MM/DD/YYYY).

The user navigates between the various editable fields using the left and right navigation arrow keys.

Notes:

- the "`password`" attribute has no effect on this input type.
- the format of the "`Default`" attribute must be MM/DD/YYYY. If "`Default`" tag is empty, today's date is displayed.

55i/57i/57iCT Default Softkeys

Position	Label	Description	URIs
5	Done	Completes the user input by submitting the programmed URI and value.	SoftKey:Submit
6	Cancel	Redisplays the previous XML object present in the phone browser. Does not appear if `LockIn` set to "yes".	SoftKey:Exit

XML Example

```
<AastraIPPhoneInputScreen type = "dateUS">
      <Title>Date US</Title>
      <Prompt>Enter date</Prompt>
      <URL>http://10.50.10.53/script.pl</URL>
      <Parameter>date</Parameter>
      <Default></Default>
```

```
<AastraIPPhoneInputScreen>
```

Resulting Screen (53i)

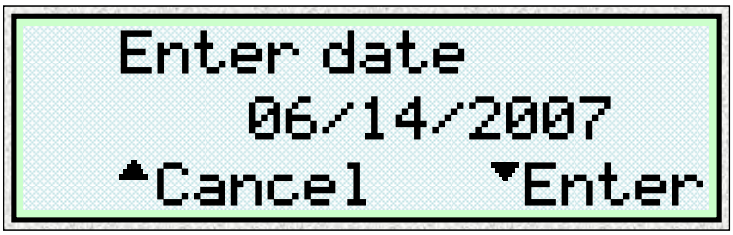


Figure 35: InputScreen “DateUS” Example (53i)

Resulting Screen (55i/57i/57iCT)

Figure 36: InputScreen “DateUS” Example (55i/57i/57iCT)

Note: In this example, when the user press “Done” on the phone after entering “06/14/2007”, the phone will call the following URL “`http://10.50.10.53/script.pl?date=06/14/2007`”.

Note: the `InputScreen` object does not perform any control on the validity of the date entered by the user.

3.6.7 Input Type: dateInt

When the input type is set to “`dateInt`”, the user can enter a date using the international format (DD/MM/YYYY).

The user navigates between the various editable fields using the left and right navigation arrow keys.

Notes:

- the “`password`” attribute has no effect on this input type.
- the format of the “`Default`” attribute must be DD/MM/YYYY. If “`Default`” tag is

empty, today's date is displayed.

55i/57i/57iCT Default Softkeys

Position	Label	Description	URIs
5	Done	Completes the user input by submitting the programmed URI and value.	SoftKey:Submit
6	Cancel	Redisplays the previous XML object present in the phone browser. Does not appear if `LockIn` set to "yes".	SoftKey:Exit

XML Example

```
<AastraIPPhoneInputScreen type = "dateInt">
      <Title>Date International</Title>
      <Prompt>Enter date</Prompt>
      <URL>http://10.50.10.53/script.pl</URL>
      <Parameter>date</Parameter>
      <Default></Default>
<AastraIPPhoneInputScreen>
```

Resulting Screen (53i)

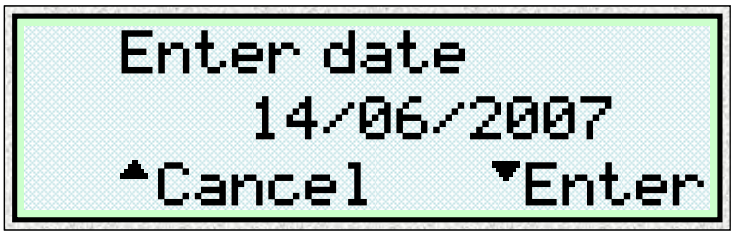


Figure 37: InputScreen "DateInt" Example (53i)

Resulting Screen (55i/57i/57iCT)

Figure 38: InputScreen "DateInt" Example (55i/57i/57iCT)

Note: In this example, when the user press “Done” on the phone after entering “14/06/2007”, the phone will call the following URL “`http://10.50.10.53/script.pl?date=14/06/2007`”.

Note: the `InputScreen` object does not perform any control on the validity of the date entered by the user.

3.7 InputScreen Object – Multiple Input fields (55i/57i/57iCT)

The InputScreen object can support up to 6 input fields, each of them having different attributes. This feature is only supported on the 55i, 57i and 57iCT phones.

The supported input types for each field are the same than the ones supported for the single input object as well as the default softkeys.

For each input field, the following attributes can be defined overriding the ones declared in the main object:

- Type
- Editable
- Password
- Prompt
- Default
- Parameter
- Custom Softkeys

Of course, only one URL can be defined to be called when the user has completed his inputs; the label in the parameter tags is appended to the address in the URL tag and sent via HTTP GET. If the Selection tag is used the value of the tag is also appended to the URL tag.

Two display modes are available for the Multiple Input AastraIPPhoneInputScreen:

- Normal: similar aspect to the single input field with a prompt and the input field on 2 separate lines, two input fields will be displayed per screen and the user will be able to scroll through them.

- Condensed: the prompt and the input field are on the same line, the prompts being right aligned on the longest prompt. Up to 5 fields will be displayed on the same screen, scrolling is not available.

XML Description

```
<AastraIPPhoneInputScreen
    type = ”IP/string/number/timeUS/timeInt/dateUS/dateInt”
    password = ”yes/no”
    editable = “yes/no”
    destroyOnExit = “yes/no”
    cancelAction = “some URI”
    Beep = “yes/no”
```

```
    Timeout = "some integer"
    LockIn = "yes/no"
    allowAnswer = "yes/no"
    defaultIndex = "some integer 1 to 6"
    displayMode = "normal/condensed"
>
      <Title wrap="yes/no">Title string</Title>
      <Prompt>Guidance for the input</Prompt>
      <URL>Target receiving the input</URL>
      <Parameter>name of the parameter added to URL</Parameter>
      <Default>Default Value (1)</Default>
      <InputField
           type = "IP/string/number
                   timeUS/timeInt/dateUS/dateInt
                   empty"
           password = "yes/no"
           editable = "yes/no"
      >
           <Prompt>Guidance for the input</Prompt>
           <URL>Target receiving the input</URL>
           <Parameter>parameter name added to URL</Parameter>
           <Default>Default Value</Default>
           <Selection>Selection</Selection>
            <!—Additional Softkey Items may be added -->
      </InputField>
<!—Additional Input fields Items may be added -->
<!—Additional Softkey Items may be added -->
</AastraIPPhoneInputScreen>
```

XML Document Objects

Document Object	Position	Type	Comments
AastraIPPhoneInputScreen	Root tag	Mandatory	Root object
Type	Root tag	Optional	Specifies the type of input, possible values are "IP", "string", "number", "timeUS", "timeInt", "dateUS", "dateInt".
Password	Root tag	Optional	Specifies if the input is masked by "*" characters. Default value is "no"
Editable	Root tag	Optional	Specifies if the user is allowed to modify the input. Default value is "yes"
destroyOnExit	Root tag	Optional	"yes/no" indicates if the object is kept or not in the phone browser after exit. If not specified, the object is kept in the browser.
cancelAction	Root tag	Optional	Defines the URI to be called when the user cancels the XML object.
Beep	Root tag	Optional	"yes" or "no" to indicate if a notification beep must be

Document Object	Position	Type	Comments
			generated by the phone.
Timeout	Root tag	Optional	Overrides the default 45 seconds timeout of the UI XML object. A Timeout set to “0” will disable the timeout feature. See section 4.4.2 for more details
LockIn	Root tag	Optional	If set to “yes”, the phone ignores all events that would cause the screen to exit without using the keys defined by the object. Default value is “no”. See section 4.4.3 for more details.
allowAnswer	Root tag	Optional	This tag applies only to the non-softkey phones (53i). If set to “yes”, the phone will display “Ignore” and “Answer” if the XML object is displayed when the phone is in the ringing state. Default value is “no”. See section 6.3 for more details.
displayMode	Root tag	Optional	If set to “normal” the input fields will be displayed with the prompt and the input field on 2 lines. If set to “condensed” both the prompt and the input field are on the same line. Default value is “normal”
defaultIndex	Root tag	Optional	Defines the field where the user will start his input amongst the multiple field inputs. Default 1.
Title	Body	Optional	Text to be used as title for the object
Wrap	Title tag	Optional	If set to “yes” the title of the object will be wrapped on 2 lines.
Prompt	Body	Optional	Text to be displayed as guidance for the user input. Is used as the default value for each input field.
URL	Body	Mandatory	URI called when user completes his input.
Parameter	Body	Optional	Name of the parameter to be added to the URL after input is complete. Is used as the default value for each input field.
Default	Body	Optional	Default value to be displayed in the input field. Is used as the

Document Object	Position	Type	Comments
			default value for each input field.
`InputField`	Body	Optional	
`Type`	InputField tag	Optional	Specifies the type of input for the field, possible values are “IP”, “string”, “number”, “timeUS”, “timeInt”, “dateUS”, “dateInt” or “empty”. Overrides the value set in the root tag for the field. An “empty” value will create one blank line in condensed mode and 2 blank lines in normal mode.
`Password`	InputField tag	Optional	Specifies if the input is masked by “*” characters. Default value is “no”. Overrides the value set in the root tag for the field.
`Editable`	InputField tag	Optional	Specifies if the user is allowed to modify the input. Default value is “yes”. Overrides the value set in the root tag for the field
`Prompt`	InputField Body	Optional	Text to be displayed as guidance for the user input. Overrides the value set in the object for the field.
`Parameter`	InputField Body	Optional	Name of the parameter to be added to the URL after input is complete. Overrides the value set in the object for the field.
`Default`	InputField Body	Optional	Default value to be displayed in the input field. Overrides the value set in the object for the field.
`Selection`	InputField Body	Optional	The content of this tag will be added when the “Submit” key is pressed while editing this field.
`SoftKey`	InputField Body	Optional	See section 4.1 for details
`SoftKey`	Body	Optional	See section 4.1 for details

Note: when the `InputField type` is set to `‘empty’`, a non editable blank line replaced the input field on the display when the XML object is in condensed mode, 2 blank lines for the normal mode. Also an empty field is a proper field; the `defaultIndex` value must consider the empty field as a plain field.

XML Example 1

```
<AastraIPPhoneInputScreen
     type="string"
     destroyOnExit="yes"
     displayMode="condensed"
>
     <Title>Restricted application</Title>
     <URL>http://myserver.com/script.php</URL>
     <Default/>
     <InputField type="empty">
     </InputField>
     <InputField type="string">
          <Prompt>Username:</Prompt>
          <Parameter>user</Parameter>
          <Selection>1</Selection>
          <SoftKey index="3">
               <Label>ABC</Label>
               <URI>SoftKey:ChangeMode</URI>
          </SoftKey>
     </InputField>
     <InputField type="number" password="yes">
          <Prompt>Password:</Prompt>
          <Parameter>passwd</Parameter>
          <Selection>2</Selection>
     </InputField>
     <SoftKey index="5">
          <Label>Done</Label>
          <URI>SoftKey:Submit</URI>
     </SoftKey>
     <SoftKey index="6">
          <Label>Exit</Label>
          <URI>SoftKey:Exit</URI>
          </SoftKey>
</AastraIPPhoneInputScreen>
```

Notes: In this example, when the user press “Done” on the phone after entering “admin” for the username and “22222” for the password, the phone will call the following URLs:

- http://myserver.com/script.php?user=admin&passwd=22222&selection=1, if the “Done” key is pressed while editing the “User name” field,
- http://myserver.com/script.php?user=admin&passwd=22222&selection=2, if the “Done” key is pressed while editing the “Password” field.

Resulting Screen

Figure 39: InputScreen multiple inputs “condensed”

XML Example 2

```
<AastraIPPhoneInputScreen
     type="string"
     destroyOnExit="yes"
>
     <Title>Date and Time</Title>
     <URL>http://myserver.com/script.php</URL>
     <Default/>
     <InputField type="dateUS">
          <Prompt>Enter Date</Prompt>
          <Parameter>date</Parameter>
     </InputField>
     <InputField type="timeUS">
          <Prompt>Enter Time</Prompt>
          <Parameter>time</Parameter>
     </InputField>
     <SoftKey index="5">
          <Label>Done</Label>
          <URI>SoftKey:Submit</URI>
     </SoftKey>
     <SoftKey index="6">
          <Label>Exit</Label>
          <URI>SoftKey:Exit</URI>
     </SoftKey>
</AastraIPPhoneInputScreen>>
```

Note: In this example, when the user press “Done” on the phone after entering “2/21/2007” for the date and “12:00:00 AM” for the time, the phone will call the following URL “http://myserver.com/script.php?date=02/21/2007&time=12:00:00AM”.

Resulting Screen

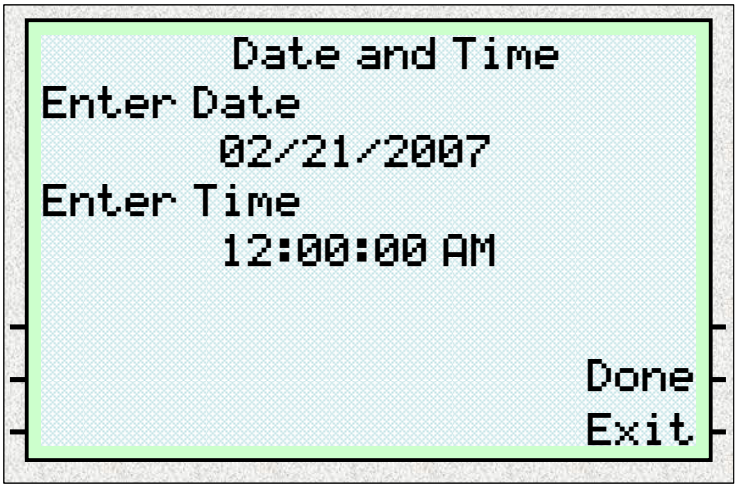


Figure 40: InputScreen multiple inputs “normal”

3.8 Directory Object (55i/57i/57i CT)

The Directory object allows the user to browse an online directory in real time. It displays an automatically numbered list of contacts. By selecting a contact with the cursor, the contact can be dialed directly by pressing the “Dial” softkey, picking up the handset, or pressing a line button. The Directory object has the optional softkeys of “Previous” and “Next” which can be linked to other XML objects.

Note: The Directory object has been kept in the current firmware for compatibility reasons with R1.3.0 but thanks to the customizable softkeys, it can be replaced by a TextMenu object with a “Dial” custom softkey.

Default Softkeys

Position	Label	Description	URIs
1	Dial	Launches a call on the first available line using the item URI as a phone number.	SoftKey:Dial
2	Previous	Optional, if pressed the phone will call the “Previous” URI as defined in the object.	Not defined
5	Next	Optional, if pressed the phone will call the “Next” URI as defined in the object.	Not defined
6	Done	Completes the user input by submitting the programmed URI and value.	SoftKey:Exit

XML Description

```
<AastraIPPhoneDirectory
  Next = “some URI”
  previous = “some URI”
  destroyOnExit = “yes/no”
  cancelAction = “some URI”
  Beep = “yes/no”
```

```
  Timeout = "some integer"
  LockIn = "yes/no"
>
      <Title wrap="yes/no">Directory Title</Title>
      <MenuItem>
            <Prompt>Contact Name</Prompt>
            <URI>number</URI>
      </MenuItem>
      <!—Additional Menu Items may be added -->
      <!—Additional Softkey Items may be added -->
</AastraIPPhoneDirectory>
```

XML Document Objects

Document Object	Position	Type	Comments
AastraIPPhoneDirectory	Root tag	Mandatory	Root object
Next	Root tag	Optional	Specifies the URI to be call for the "Next" softkey
Previous	Root tag	Optional	Specifies the URI to be call for the "Previous" softkey
destroyOnExit	Root tag	Optional	"yes/no" indicates if the object is kept or not in the phone browser after exit. If not specified, the object is kept in the browser.
cancelAction	Root tag	Optional	Defines the URI to be called when the user cancels the XML object.
Beep	Root tag	Optional	"yes" or "no" to indicate if a notification beep must be generated by the phone.
Timeout	Root tag	Optional	Overrides the default 45 seconds timeout of the UI XML object. A Timeout set to "0" will disable the timeout feature. See section 4.4.2 for more details
LockIn	Root tag	Optional	If set to "yes", the phone ignores all events that would cause the screen to exit without using the keys defined by the object. Default value is "no". See section 4.4.3 for more details.
Title	Body	Mandatory	Text to be used as title for the object
wrap	Title tag	Optional	If set to "yes" the title of the object will be wrapped on 2 lines.
MenuItem		Mandatory	Choice Item (up to 15 instances)
Prompt		Mandatory	Label of the item
URI		Mandatory	Phone number or IP address to be dialed

Document Object	Position	Type	Comments
SoftKey	Body	Optional	See section 3.9 for details

XML Example

```
<AastraIPPhoneDirectory
  next="http://myserver.com/more.php"
  previous=http://myserver.com/back.xml
>
      <Title>My Directory</Title>
      <MenuItem>
            <Prompt>John Doe 1</Prompt>
            <URI> 10.50.10.49</URI>
      </MenuItem>
      <MenuItem>
            <Prompt>John Doe 2</Prompt>
            <URI>4326</URI>
      </MenuItem>
      <MenuItem>
            <Prompt>John Doe 3</Prompt>
            <URI>9982691234</URI>
      </MenuItem>
</AastraIPPhoneDirectory
```

Resulting Screen

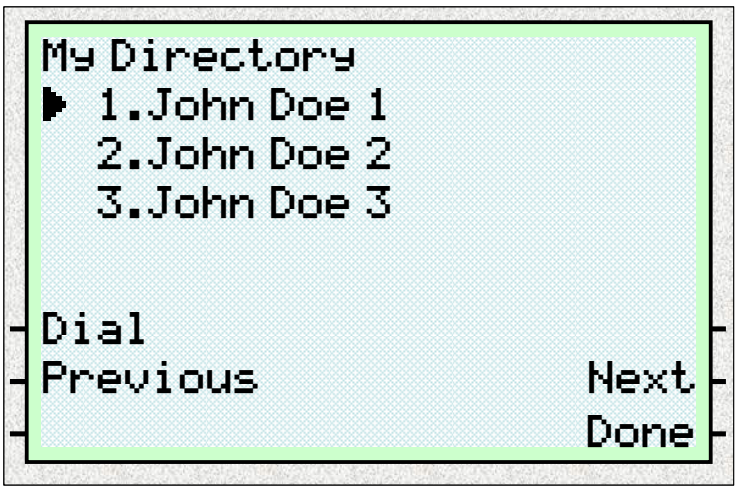


Figure 41: Directory Example

3.9 PhoneStatus Object (All models)

The AastraIPPhoneStatus object provides the ability to display a status message on a single designated line on the phone's idle screen when XML information is pushed from the servers.

The 55i/57i/57i CT phones display messages on the second line in the phone window (where "No Service" would display if there was no service. If there is no service on the phone, the "No Service" message overrides the XML object message). The 53i phone display messages on the second line. Long messages that are wider then the phone screen get truncated.

If the phone receives multiple messages, the first message received displays first and the remaining messages scroll consecutively one at a time.

The AastraIPPhoneStatus object supports 2 modes for the messages to be displayed:

- Normal mode: messages remain displayed until they are removed (by the server) or the phone reboots.
- Alert mode: Messages are displayed on top of the existing messages only for a limited time (3 seconds by default)

The `AastraIPPhoneStatus` object feature is always enabled.

Note: You can set the amount of time, in seconds, that a message displays to the phone before scrolling to the next message (See *Scroll delay* below).

Default Softkeys

Not Applicable, this object has no predefined softkey as it impacts the idle screen of the Aastra SIP Phone

XML Description

```
<AastraIPPhoneStatus
   Beep = "yes/no"
   triggerDestroyOnExit = "yes/no"
>
      <Session>Session ID</Session>
      <Message
       Index = "index"
       Type = "alert"
       Timeout = "timeout"
       >Message</Message>
      <!—Additional Message Items may be added -->
</AastraIPPhoneStatus>
```

Notes:

- The *Session ID* must be unique to the application sending the XML object to the phone. It is up to the application to generate that session ID, which does not have to be limited to just numbers. It could be a combination of letters and numbers. There could only be one Session tag per PhoneStatusMsg object. If the `Session` tag is not provided, the phone assumes a default value (0) for it; this can be used if you don't have multiple applications displaying messages on the idle screen.
- The `type="alert"` tag indicates the alert mode if not specified the message is displayed in the normal mode.

Scroll Delay

The Scroll delay can be configured via the configuration files and the Aastra Web UI using the following parameters:

- `xml status scroll delay` (via configuration files)
- Status Scroll Delay (in seconds) (via the Aastra Web UI see chapter 7.6)

XML Document Objects

Document Object	Position	Type	Comments
`AastraIPPhoneStatus`	Root tag	Mandatory	Root object
`Beep`	Root tag	Optional	“yes” or “no” to indicate if a notification beep must be generated by the phone.
`triggerDestroyOnExit`	Root tag	Optional	If the XML object is sent as an answer to a UI object, setting this parameter to “yes” will trigger the exit of the calling object as if it was a UI object. See section 4.4.4 for more details.
`Session`	Body	Optional	Session ID used to identify the application displaying message. It allows message change and message reset.
`Message`	Body	Mandatory	Message to be displayed or empty to reset the message.
`index`	Message tag	Mandatory	Index of the message for the session, the index starts at 0.
`type`	Message tag	Optional	Type of message, only “alert” is supported. If not specified the message stays on the screen until it is reset by an empty message or after a phone reboot.
`Timeout`	Message tag	Optional	Timeout of the “alert” message, overrides the 3s default value.
`SoftKey`	Body	Optional	See section 4.1 for details

XML Example

```
<AastraIPPhoneStatus Beep=”yes”>
      <Session>abc12345</Session>
      <Message index=”0”>Message 1 displayed</Message>
      <Message index=”1” type=”alert” Timeout=”5”>Alert
displayed</Message>
</AastraIPPhoneStatus>
```

Resulting Screen (53i)

Figure 42: PhoneStatus Example (53i)

Resulting Screen (55i/57i/57iCT)

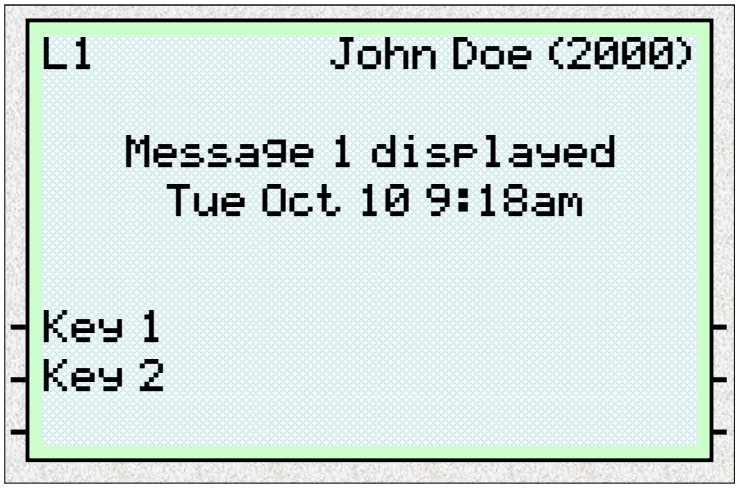


Figure 43: PhoneStatus Example (55i/57i/57iCT)

Note: The `PhoneStatusMsg` object can also be used to remove status messages from the display using the same `SessionID` and the same `index`. This can be accomplished by setting an empty tag for the `Message` tag. For example, here is the XML object to remove the previous messages (as the second message was in alert mode it does mot remain on the display so no need to remove it).

```
<AastraIPPhoneStatus>
      <Session>abc12345</Session>
      <Message index="0"/>
<AastraIPPhoneStatus/>
```

3.10 PhoneExecute Object (All models)

The `PhoneExecute` object allows an external application to ask the phone to execute a sequence of local actions using a URI.

The actions can be:

- Any supported HTTP(s) URL
    - http://myserver.com/myscript.pl
    - Dial:XXXXX

- o Led: XXXXXX=on/off/slowflash/fastflash see chapter 4.3 for more details on the LED control.

- Phone Reboot (URI="Command: Reset"), the phone will restart (if the phone is idle) and process the complete boot sequence (configuration, language packs, firmware…)
- Phone Fast Reboot (URI="Command: FastReboot"), the phone will restart (if the phone is idle) but will reduce the boot sequence as it will not check for new firmware and will only download language packs if there is a change in supported languages
- Do nothing (URI="")

More actions will be implemented in future firmware versions.

In order to prevent a pushed dial uri from putting an active call on hold the PhoneExecute object supports an optional tag, `interruptCall`. By default, the attribute will allow the current call to be interrupted. To prevent this set the attribute to "no".

Default Softkeys

Not Applicable, this object has no predefined softkey.

XML Description

```
<AastraIPPhoneExecute
   Beep = "yes/no"
   triggerDestroyOnExit="yes/no"
>
      <ExecuteItem URI="URI" interruptCall="yes/no"/>
      <!–Additional ExecuteItems may be added -->
</AastraIPPhoneExecute>
```

XML Document Objects

Document Object	Position	Type	Comments
AastraIPPhoneExecute	Root tag	Mandatory	Root object
Beep	Root tag	Optional	"yes" or "no" to indicate if a notification beep must be generated by the phone.
triggerDestroyOnExit	Root tag	Optional	If the XML object is sent as an answer to a UI object, setting this parameter to "yes" will trigger the exit of the calling object as if it was a UI object. See section 4.4.4 for more details.
ExecuteItem	Body	Optional	Tag for the action to be executed
URI	ExecuteItem tag	Optional	URI describing the action to be executed.
interruptCall	ExecuteItem tag	Optional	When the URI is a dial command, if this tags is set to 'no' an existing call will not be put on hold.

XML Example

```
<AastraIPPhoneExecute>
      <ExecuteItem URI=http://myserver.com/myscript.php/>
      <ExecuteItem URI="Dial: 12345" interruptCall="no"/>
```

```
      <ExecuteItem URI="Led: softkey1=fastflash"/>
      <ExecuteItem URI="Command: Reset"/>
</AastraIPPhoneExecute>
```

This example will make the phone execute 3 actions:

- Do an HTTP GET to myserver.com/myscript.php
- Dial 12345 without putting any current call on hold
- Change the status of the LED associated to the softkey1 to ON.
- Reset the phone

Notes:

- the "do nothing" can be used when an application needs to display nothing as an answer to a HTTP GET.
- you must be careful when you use the "Dial:" URI as the state of phone is unknown at the time of the XML GET.
- the FastReboot command will speed up the boot process of the phone which may be useful in the self-configuration application.

3.11 PhoneConfiguration Object (All models)

The `PhoneConfiguration` object allows an external application to modify the phone configuration dynamically. The configuration parameters are the ones that are used in the configuration files (Aastra.cfg and <MAC>.cfg) detailed in the administrator guide.

The configuration parameters changed by the use of the `PhoneConfiguration` are not saved locally by the phone (*local.cfg* file), when the phone reboots, the new parameters sent using the PhoneConfiguration XML object are lost.

So in order to keep the changes, the XML application should also update the configuration files or update them after a reboot (using an action uri startup for instance).

The phone will override its current parameters only if they have not been set or if they have been set by the configuration files, the parameters set locally (using the TUI or WUI) can not be overridden.

The number of parameters to be sent in a single `Phoneconfiguration` object is only limited by the overall size of the XML object (10000 bytes). Practically 30 parameters can be sent in a single object.

Note: Not all the configuration parameters are dynamic; specifically the SIP configuration and the network parameters are static and need a reboot.

The list of the dynamic configuration parameters is detailed in section 13.

Default Softkeys

Not Applicable, this object is a non UI XML object and has no predefined softkey.

XML Description

```
<AastraIPPhoneConfiguration
   Beep = "yes/no"
   triggerDestroyOnExit = "yes/no"
```

```
>
      <ConfigurationItem>
            <Parameter>parameter</Parameter>
            <Value>value</Value>
      </ConfigurationItem>
      <!—Additional ConfigurationItems may be added -->
</AastraIPPhoneConfiguration>
```

XML Document Objects

Document Object	Position	Type	Comments
AastraIPPhoneConfiguration	Root tag	Mandatory	Root object
Beep	Root tag	Optional	“yes” or “no” to indicate if a notification beep must be generated by the phone.
triggerDestroyOnExit	Root tag	Optional	In case the XML object is sent as an answer to a UI object, setting this parameter to “yes” will trigger the exit of the calling object as if it was a UI object. See section 4.4.4 for more details.
ConfigurationItem	Body	Optional	Tag for the configuration change
Parameter	ConfigurationItem tag	Mandatory	Configuration parameter to be changed
Value	ConfigurationItem tag	Mandatory	New value of the configuration parameter.

XML Example

```
<AastraIPPhoneConfiguration>
      <ConfigurationItem>
            <Parameter>softkey5 label</Parameter>
            <Value>New Label</Value>
      </ConfigurationItem>
</AastraIPPhoneConfiguration>
```

This example will change the label of the softkey 5 to “New Label” on a 55i/57i and 57i CT.

4 XML extensions

4.1 Graphics (55i/57i/57i CT)

4.1.1 Images

The `image` tag is used by the ImageScreen and ImageMenu XML objects.

Format

Two hex characters map to one byte of pixel data, where each bit represents a pixel. The image data describes the bitmap from left to right and top to bottom. The data is padded on an 8-bit boundary, so if the height and width do no match the pixel information, then the image will not display correctly.

The entire LCD on the 5 series IP phones can be used to display images.

Notes:

- with firmware 2.1.0, the size of the images to be displayed is limited to 144x40 pixels.
- Aastra provides a Microsoft Windows tool called “bitmapconverter” able to convert any black and white Windows bitmap to the hexadecimal string to be used with Aastra graphical XML objects. The syntax under Windows command line interface is “bitmapconverter XXXXX.bmp”, the output is the result.

Screen resolution

The 55i screen has a resolution of 144 by 75 pixels.

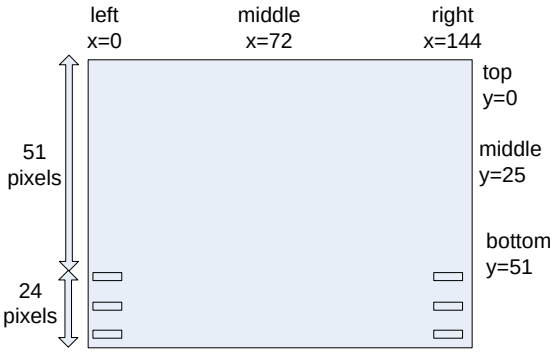


Figure 44: Aastra 55i screen.

Similarly, the 57i/57i CT has a resolution of 144 by 128 pixels. The center of this display is at (72, 64).

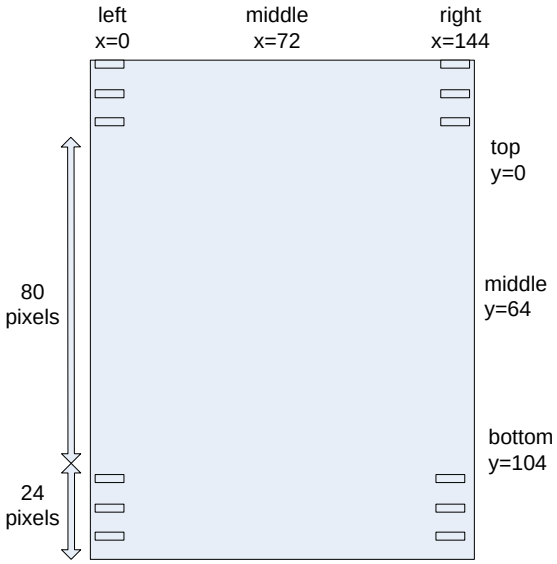


Figure 45: Aastra 57i/57i CT screen

4.1.2 Icons

On top of images, the 55i, 57i and 57i CT support icons in some XML objects:

- TextMenu
- Custom softkeys

Format

The icons use the same encoding format than the images: two hex characters map to one byte of pixel data, where each bit represents a pixel, the icon data describes the icon from left to right.

Icons are 8 pixels high and a multiple of 6 pixels wide.

As the font used by the phone is 7 pixels high, in order to align the icons with the labels, it is recommended to use icons with the first row of pixel (from the bottom) empty. Also it is recommended to leave the last column empty to leave a space between the icon and the label.

In the following example which is a single cell icon, it is recommended to use 5x7 icons.

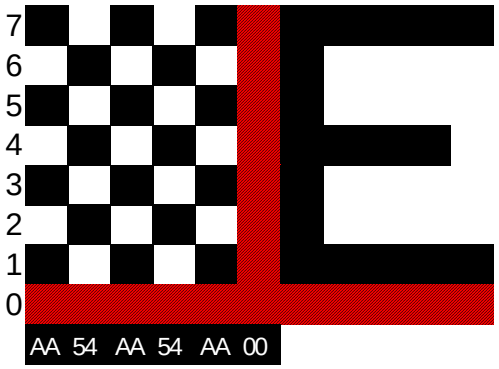


Figure 46: One cell icon

For a two cell icon, the recommended size is 11x7.

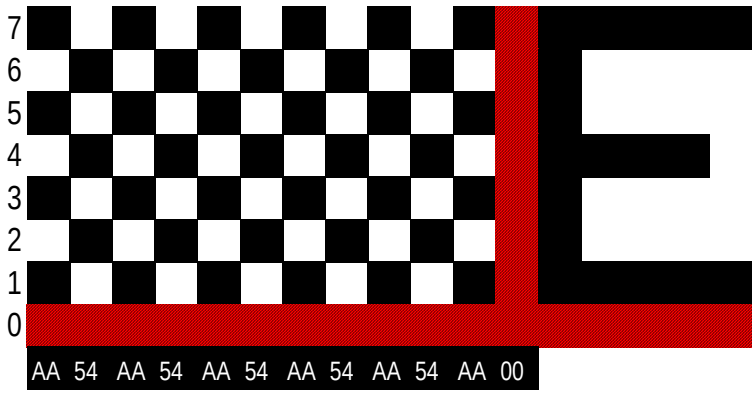


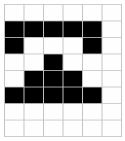
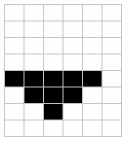
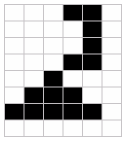
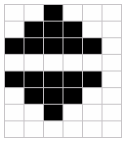
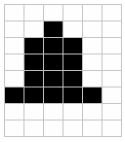
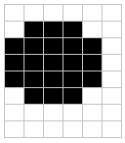
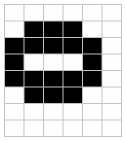
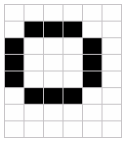
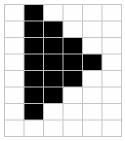
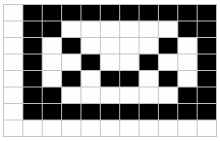
Figure 47: Two cell icon

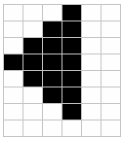
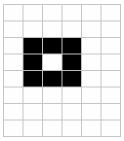
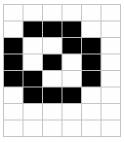
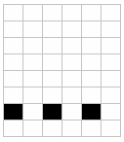
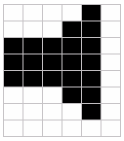
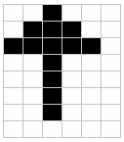
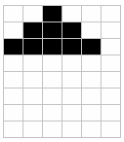
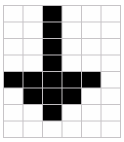
3 cells: 17x7…

Predefined icons

Some predefined icons are available in the phone. To use the predefined icons you must set the `Icon` tag to Icon:IconName where IconName is the name of the predefined icon.

Here is the list of the available icons.

Name	Icon	Name	Icon
PhoneOnHook		ArrowDown	
PhoneOffHook		ArrowsUpAndDown	
PhoneRinging		FilledCircle	
DND		EmptyCircle	
ArrowRight		Envelope	

Name	Icon	Name	Icon
ArrowLeft		Square	
Prohibit		Ellipse	
Speaker		TailArrowUp	
ArrowUp		TailArrowDown	

Icons declaration

Icons are identified by an index in the XML objects; this index refers to the `IconList` tag which must be present each an icon is used.

```
<IconList>
      <Icon index = "iconindex">Icon:Iconname or Hex Icon</Icon>
      <Icon index = "iconindex">Icon:Iconname or Hex Icon</Icon>
      <!–As many as used in the object definition -->
</IconList>
```

Note: in the TextMenu object which can have icons in the `MenuItems` and in the custom Softkeys, all the icons must be defined in a single `IconList` tag.

4.2 Customizable Softkeys (55i/57i/57i CT only)

The Softkey object can be used to override the default softkeys in each of the XML objects. It allows developers to link arbitrary URIs to keys in the XML screens and invoke softkey behavior native to each XML screen type.

Notes:

- custom softkeys are only available for the UI XML objects.
- when you use custom softkeys, the default softkeys of the XML object are not displayed anymore. This means they have to be recreated as custom softkeys.

XML Description

```
<SoftKey index = "1-6" icon = "iconindex">
      <Label>Text</Label>
      <URI>http://someserver/somepage OR SoftKey:someaction</URI>
</Softkey>
<IconList>
      <Icon index = "iconindex">Icon:Iconname or Hex Icon</Icon>
      <Icon index = "iconindex">Icon:Iconname or Hex Icon</Icon>
      <!—As many as used in the softkey definition -->
</IconList>
```

XML Document Objects

Document Object	Position	Type	Comments
SoftKey	Body	Mandatory	Softkey Root object (up to 6)
Index	SoftKey Root tag	Mandatory	Indicates the softkey number
Icon	SoftKey Root tag	Optional	Index of the icon to be used from the icon list
Label	SoftKey Body	Mandatory	Label of the softkey
URI	SoftKey Body	Mandatory	URI called if the softkey is pressed
IconList	Body	Optional	Icon root object
Icon	Iconlist Body	Optional	Icon definition see chapter 4.1.2 for more details.
index	Icon tag	Optional	Icon index

You can define up to six softkeys before the closing tag of any object. The following example illustrates the use of the softkey XML element used with the Text Menu object, and the resulting screen output.

Available object commands

The following softkey functionality is available to the developer for the purpose of reordering or preserving the default functionality of the XML screens.

- "SoftKey:Select" is available for AastraIPPhoneTextMenu only and calls the URI tag of the selected MenuItem.
- "SoftKey:Exit" is available for all UI XML objects and redisplays the previous XML object present in the phone browser. Does not appear if LockIn set to "yes".
- "SoftKey:Dial" is available to screens that allow input. The dial string for the "Dial" function is taken from the menu items URI on the Menu Screen, and from the editor field input on the Input Screen.
- "SoftKey:Dial2" is available only for the AastraIPPhoneTextMenu object and will dial the number set by the "Dial" tag.
- "SoftKey:Submit" is available only for the AastraIPPhoneInputScreen object, it completes the user input by submitting the programmed URI (URL tag) and value (Parameter tag).

- "SoftKey:BackSpace" is available only for the AastraIPPhoneInputScreen object, it deletes the character placed before the cursor.

- "SoftKey:NextSpace" is available only for the AastraIPPhoneInputScreen object, it inserts a "space" character at the cursor position.

- "SoftKey:Dot" is available only for the AastraIPPhoneInputScreen object, it inserts a "." character at the cursor position.

- "SoftKey:ChangeMode" is available only for the AastraIPPhoneInputScreen object, it allows a toggle between lower case, upper case and digit inputs.

- "SoftKey:Answer" is available for all UI XML objects and allows the user to answer a call. This softkey will only be displayed if it is created while the phone is in an incoming state and diseappears when the call is answered or ignored.

- "SoftKey:Ignore" is available for all UI XML objects and allows the user to ignore a call. This softkey will only be displayed if it is created while the phone is in an incoming state and diseappears when the call is answered or ignored.

- "SoftKey:SymbolList="XYZ"" is available only for the AastraIPPhoneInputScreen object, it allows an input of a custom list of characters at the cursor position.

  The format is:

```
<SoftKey index="1">
      <Label>Symbols</Label>
      <URI>SoftKey:SymbolList="@."</URI>
</SoftKey>
```

  The content of the Symbol List must be encapsulated by quotes. Note that there can be multiple SymbolList softkeys with different list of symbols. There are some special characters that need to be encoded due to XML limitations (see chapter 2.6 for the encoded value).

SoftKey	TextScreen	TextMenu	InputScreen	Directory
Select		X		
Exit	X	X	X	X
Dial		X	X	X
Dial2		X		
Submit			X	
BackSpace			X	
NextSpace			X	
Dot			X	
ChangeMode			X	
Answer[1]	X	X	X	X
Ignore[1]	X	X	X	X
SymbolList			X	

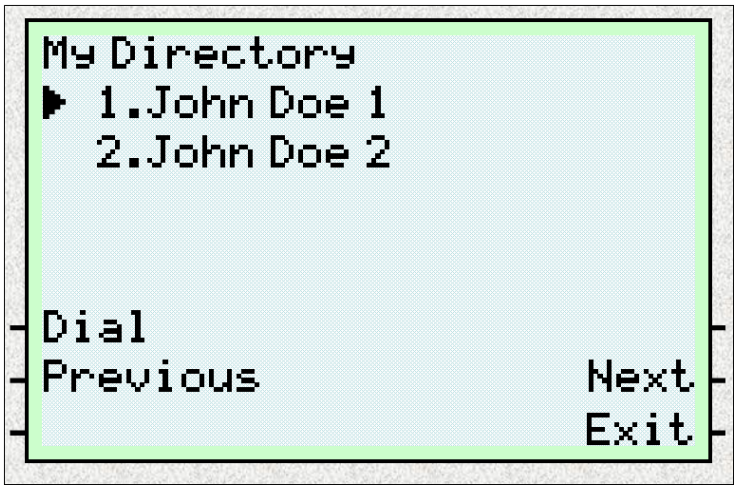
[1] This softkey will only be displayed if it is created while the phone is in an incoming state. Pressing it will ignore the incoming call without disturbing the current XML application. Once the call is answered or ignored, this softkey disappears automatically.

SoftKey	Formatted TextScreen	ImageScreen	ImageMenu
Select			
Exit	X	X	X
Dial			
Dial2			
Submit			
BackSpace			
NextSpace			
Dot			
ChangeMode			
Answer	X	X	X
Ignore	X	X	X
SoftKey List			

XML Example1

```
<AastraIPPhoneTextMenu>
      <Title> My Directory </Title>
      <MenuItem>
            <Prompt>John Doe 1</Prompt>
            <URI>10.50.10.49</URI>
      </MenuItem>
      <MenuItem>
            <Prompt>John Doe 2</Prompt>
            <URI>200</URI>
      </MenuItem>
      <SoftKey index = “1”>
            <Label>Dial</Label>
            <URI>SoftKey:Dial</URI>
      </Softkey>
      <SoftKey index = “2”>
            <Label>Previous</Label>
            <URI>http://someserver/somepage.xml</URI>
      </Softkey>
      <SoftKey index = “5”>
            <Label>Next</Label>
            <URI>http://someserver/somepage.xml</URI>
      </Softkey>
      <SoftKey index = “6”>
            <Label>Exit</Label>
            <URI>SoftKey:Exit</URI>
      </Softkey>
</AastraIPPhoneTextMenu>
```

Resulting Screen



XML Example2

```
<AastraIPPhoneInputScreen type="string">
      <Title>Title</Title>
      <Prompt>Enter email address</Prompt>
      <URL>http://myserver.com/script.php</URL>
      <Parameter>email</Parameter>
      <Default></Default>
      <SoftKey index="1">
            <Label>Backspace</Label>
            <URI>SoftKey:BackSpace</URI>
      </SoftKey>
      <SoftKey index="2">
            <Label>Symbols</Label>
            <URI>SoftKey:SymbolList="@."</URI>
      </SoftKey>
      <SoftKey index="5">
            <Label>Done</Label>
            <URI>SoftKey:Submit</URI>
      </SoftKey>
      <SoftKey index="6">
            <Label>Exit</Label>
            <URI>SoftKey:Exit</URI>
      </SoftKey>
</AastraIPPhoneInputScreen>
```

Resulting Screen

4.3 LED control

You can control the status of the LED associated to a phone key **as long as the key is configured as an XML key**. This can be done using the universal URI

"Led: XXXXX=on/off/fastflash/slowflash"

Where XXXXX represents the key you want to modify the LED.

Note: the LED state is lost after a phone reboot.

The supported LED states are:

- on, the LED is steady on
- off, the LED is steady off
- slowflash, the LED blinks at a slow pace
- fastflash, the LED blinks at a fast pace

Keys supported on a 53i

- prgkey1 to prgkey6

Keys supported on a 55i

- prgkey1 to prgkey6
- softkey1 to softkey20

Keys supported on a 57i/57i CT

- topsoftkey1 to topsoftkey10
- softkey1 to softkey20

Keys supported on an expansion module (53i/55i/57i/57iCT)

- 536M: expmodX key1 to expmodX key36
- 560M: expmodX key1 to expmodX key60

- Where X is the expansion module number (1 to 3)

XML Example

```
<AastraIPPhoneExecute>
     <ExecuteItem URI="Led: topsoftkey10=on">
     <ExecuteItem URI="Led: softkey1=off">
     <ExecuteItem URI="Led: prgkey2=fastflash">
     <ExecuteItem URI="Led: expmod2 key20=slowflash">
</AastraIPPhoneExecute>
```

4.4 Special attributes

4.4.1 Beep

You can enable or disable a `Beep` option in all the Aastra XML objects via the Beep attribute in the root tag. When the phone receives an XML object, the `Beep` notifies the user that an object is being displayed.

This attribute is optional. If the `Beep` attribute is set to "yes" (i.e. `Beep`="yes") then it is an indication to the phone to sound a beep when it receives the object. If the `Beep` attribute is set to "no" (i.e. `Beep`="no") or not present, then the default behavior is no beep is heard when the object arrives to the phone.

The `Beep` option can also be enabled or disabled via the configuration files and the Aastra Web UI using the following parameters:

- `xml beep notification` (via configuration files)
- XML Beep Support (via the Aastra Web UI see chapter 7.5)

The value set in the configuration files and Aastra Web UI override the attribute you specify in the XML object.

For example, if an `AastraIPPhoneStatus` object has the attribute of Beep="yes", and you uncheck (disable) the "XML Beep Support" in the Aastra Web UI, the phone does not beep when it receives an `AastraIPPhoneStatus` object.

4.4.2 Timeout

The `Timeout` attribute is an optional root tag attribute for all of the current UI XML objects. When the phone receives an XML object with this attribute set it will override the default 45 seconds timeout specified for custom applications.

Setting `Timeout` to "0" will disable the timeout feature.

This timeout is reset by button presses.

Note: A timeout that is shorter than a Refresh header time will be ignored.

4.4.3 LockIn

The `LockIn` attribute is an optional root tag attribute for all of the current UI XML objects. This attribute allows the XML designer to specify that a screen can not be cancelled.

When a phone receives an UI XML object with the attribute set to "yes" it ignores all events that would cause the screen to exit without using the keys defined by the object.

Note: An incoming call still cancels the “locked” screen.

Setting `LockIn` will disable the default timeout feature (45 seconds) unless the `Timeout` attribute is also set in the root tag.

On the InputScreen object, setting the LockIn attribute to “yes” will deactivate the “Cancel” key. It is a way to force an input.

4.4.4 triggerDestroyOnExit

The triggerDestroyOnExit is an optional root tag attribute for all of the current non-UI XML objects (PhoneStatus, PhoneExecute and PhoneConfiguration). Its default value is “no”.

By default, if a UI XML object gets a non-UI XML object as an answer to an exit URI (“Select” on a TextMenu or “Done” on a InputScreen…) the UI XML object is not destroyed and stays on the phone display even if its `destroyOnExit` tag is set to “yes”.

By setting `triggerDestroyOnExit` to “yes”, the previous UI XML object is destroyed if its `destroyOnExit` tag is set to “yes”.

4.5 TextMenu user selection (55i/57i/57iCT)

It is also possible to send information related to a user’s choice in a Text Menu.

“Select” and “Dial” system keys use the URI passed in the `MenuItem` attribute, a custom key might need to know the user selection in the TextMenu, and this is the purpose of the **`Selection`** attribute.

When a user accesses an arbitrary URI via a custom softkey, the Text Menu will send along a bit of information regarding the user’s currently selected option using an optional XML attribute **`Selection`**.

XML Example

```
<AastraIPPhoneTextMenu destroyOnExit = ”yes”>
      <Title>Parameter Tester</Title>
      <MenuItem>
            <Prompt>First Selection</Prompt>
            <URI>http://someserver/somepage.xml</URI>
            <Selection>dataToAppend</Selection>
      </MenuItem>
      <SoftKey index = “1”>
            <Label>Custom Key</Label>
            <URI>http://someotherserver/someotherpage.xml</URI>
      </SoftKey>
</AastraIPPhoneTextMenu>
```

When the user selects item 1 and presses softkey 1, the URI requested is
http://someotherserver/someotherpage.xml?selection=dataToAppend

Notes:

- If a “?” already exists in the URI, then a “&” is used to separate the parameters. Note the parameter name “selection” is automatic. If the `Selection` attribute is omitted, then nothing extra is appended to the URI.
- the `Selection` attribute can be more than one parameter, for instance, the following

`Selection` attribute is valid `<Selection>200&action=set</Selection>` (don't forget to escape encode the attribute for the "&").

4.6 "Select" and "Dial" in the same TextMenu object

The only way to have a "Select" and "Dial" keys in the same text menu, the only option is to use "Dial" as a system command and mimic "Select" with a custom key.

```
<AastraIPPhoneTextMenu destroyOnExit = "yes">
      <Title>Dial and Select</Title>
      <MenuItem>
            <Prompt>First Selection</Prompt>
            <URI>200</URI>
            <Selection>200</Selection>
      </MenuItem>
      <MenuItem>
            <Prompt>Second Selection</Prompt>
            <URI>201</URI>
            <Selection>201</Selection>
      </MenuItem>
      <SoftKey index = "1">
            <Label>Dial</Label>
            <URI>SoftKey:Dial</URI>
      </SoftKey>
      <SoftKey index = "2">
            <Label>Select</Label>
            <URI>http://myserver.com/script.php</URI>
      </SoftKey>
</AastraIPPhoneTextMenu>
```

When the user selects item 1 and presses

- Softkey 1, the phone dials 200
- Softkey 2, the requested URI requested is http://myserver.com/script.php?selection=200

4.7 "Select", "Dial" and "Dial2" behavior in a TextMenu object

The following table describes what actions will be performed for each possible combination of items and actions in a `<MenuItem>`.

MenuItem Content	Select Softkey	Offhook	Dial2 Softkey	Dial Softkey
<URI>http://XYZ.</URI>	GET(http://XYZ)	-	-	Dial(http://XYZ)
<Dial>123</Dial>	-	Dial(123)	Dial(123)	-
<URI>http://XYZ</URI> <Dial>123</Dial>	GET(http://XYZ)	Dial(123)	Dial(123)	Dial(http://XYZ)
<URI>Dial:456</URI> <Dial>123</Dial>	Dial(456)	Dial(123)	Dial(123)	Dial(Dial:456)

Note: the main purpose of the "Dial" tag in the TextMenu is to allow the Aastra SIP phones which do not support custom softkeys (33i and 53i) to dial a number going off-hook.

4.8 Refresh of an XML page

All XML screen objects have the ability to be refreshed. This is accomplished by adding a Refresh setting to the HTTP headers. This setting comprises two parameters:

- a time in seconds
- a URL.

The Refresh setting is set by the XML application and it is up to the application to decide which objects they want to refresh. So, it is an optional setting.

If the setting appears, then both parameters must be set for the Refresh to work.

The format of the HTTP header is:

```
Refresh: timeout; url=page to load
```

Example

```
Refresh: 3; url=http://10.50.10.140/update.xml
```

All of the parameters are mandatory.

With php you will have to use the `header` command in your script.

```
header("Refresh: 1; url= http://10.50.10.140/update.xml");
```

4.9 HTTP headers format for a phone HTTP GET

When the Aastra SIP performs a GET to an http server, it sets multiple HTTP headers:

- HTTP_USER_AGENT providing information on the characteristics of the phone (type, MAC address, firmware),
- HTTP_X_AASTRA_EXPMOD1 providing information on the expansion module 1 if present,
- HTTP_X_AASTRA_EXPMOD2 providing information on the expansion module 2 if present,
- HTTP_X_AASTRA_EXPMOD3 providing information on the expansion module 3 if present,
- HTTP_ACCEPT_LANGUAGE providing information on the language used on the phone.

4.9.1 HTTP_USER_AGENT

The HTTP_USER_AGENT header includes:

- The type of phone (53i, 55i, 57i, 57i CT))

- The MAC address of the phone
- The firmware version of the phone

Usually the HTTP server is able to retrieve the User Agent parameters. The Aastra SIP phones send the following string for the User Agent.

`Phone_Model[space]MAC:-XX-XX-XX-XX-XX-XX[space]V:Version`

This allows the application to adapt to the type of phone it is dealing with but also allows to design a single application to provide XML (for the phones) and HTML (for a web browser).

The following source code is a php example that can be used to decode the HTTP_USER_AGENT header sent by an Aastra phone.

```
function Aastra_decode_HTTP_header()
{
$user_agent=$_SERVER["HTTP_USER_AGENT"];
if(stristr($user_agent,"Aastra"))
      {
      $value=preg_split("/ MAC:/",$user_agent);
      $end=preg_split("/ /",$value[1]);
      $value[1]=preg_replace("/\-/","",$end[0]);
      $value[2]=preg_replace("/V:/","",$end[1]);
      }
else
      {
      $value[0]="Unknown";
      $value[1]="NA";
      $value[2]="NA";
      }
$header['model']=$value[0];
$header['mac']=$value[1];
$header['firmware']=$value[2];

return($header);
}
```

This function returns an array with

- Array['model] is the phone model
    - "Aastra53i",
    - "Aastra55i",
    - "Aastra57i"
    - "Aastra57iCT"
- Array['mac'] is the MAC address
- Array['firmware'] is the firmware version

Example of execution

- Array['model'] => Aastra57i
- Array['mac'] => 00085D031C81
- Array['firmware'] => 2.1.0.1073-SIP

4.9.2 HTTP_X_AASTRA_EXPMOD

If the phone has expansion modules (53i, 55i, 57i or 57i CT) connected, the phone sends this information in proprietary http headers:

- **HTTP_X_AASTRA_EXPMOD1**
- **HTTP_X_AASTRA_EXPMOD2**
- **HTTP_X_AASTRA_EXPMOD3**

The header is sent only if an expansion module is present at the given position, the possible values are:

- 536M for the 36 keys paper labeled expansion module (53i, 55i, 57i, 57i CT)
- 560M for the 60 keys self labeled expansion module (55i, 57i, 57i CT)

The following source code is a php example that can be used to decode the HTTP_X_AASTRA_EXPMOD header sent by an Aastra phone.

```
function Aastra_decode_HTTP_X_Aastra()
{
$header['module1']=$_SERVER["HTTP_X_AASTRA_EXPMOD1"];
$header['module2']=$_SERVER["HTTP_X_AASTRA_EXPMOD2"];
$header['module3']=$_SERVER["HTTP_X_AASTRA_EXPMOD3"];

return($header);
}
```

Example of execution

An Aastra 57i with one 560M in position 1 and a 536M in position 2.

- Array['module1'] => 560M
- Array['module2'] => 536M
- Array['module3'] => Empty string

4.9.3 HTTP_ACCEPT_LANGUAGE

The phone also sends the standard **HTTP_ACCEPT_LANGUAGE** in the headers request which describes the language of the phone using 2 characters.

To access it using php the syntax is `$_SERVER["HTTP_ACCEPT_LANGUAGE"])`

Example

- `echo $_SERVER["HTTP_ACCEPT_LANGUAGE"]`
- `returns "en"`

4.10 Some development guidelines

There are some simple rules that you should follow when you develop XML applications for the Aastra SIP phones.

- Don't forget the "Exit" key when you use custom softkeys
- Place custom softkeys as they are for the standard objects, also it is better to use the same labels. The Exit key should preferably be placed in position 6.

- Setting **`destroyOnExit`** attribute to “yes” is preferable when you write complex applications as it is the only way to properly control the pages that are stacked in the phone.

- Avoid using the Directory Object; TextMenu object with custom keys is much more flexible.

- If you want to access data from the internet, it is preferable to use a RSS feed than Web scraping as Web sites frequently change their interface.

- For XML applications using data from the Internet, it they are not real time you might consider to “cache” the data on the server and update them on a regular basis (a weekly update is necessary for Movie schedules but a 10 minutes update might be necessary for a weather application).

- As the XML objects are limited to 10000 bytes, you might sometime hit this limit for a complex `TextMenu` with a lot of custom keys and custom icons. Don’t forget to use the base attribute in the `MenuItem` object as it is a good way to reduce the size of the object.

5 URL Format and Variables

5.1 URL format

To access to an XML application the phone performs a HTTP (or HTTPS) GET on a URL. The supported syntax is:

http://host[:port]/dir/file

or

https://host[:port]/dir/file

where:

- host is the hostname or IP address of the Web server supporting the XML application
- port is the port number the phones is using for his HTTP request.

Note: If the port is not specified, the phone uses port 80.

The phone also supports the following URL:

- `Dial: XXXXX` to have the phone dial XXXXX
- `Led: XXXXX=on/off/slowflash/fastflash` to change the status the LED associated to the XXXXX key if the key is configured as an XML key. See chapter 4.3 for more details on the LED control.

5.2 URL Variables

The phones support system variables in the HTTP URL to pass dynamic data to the XML applications.

The variables are in the form $$VARIABLENAME$$ and the following variables are supported:

- $$SIPUSERNAME$$ line user name
- $$DISPLAYNAME$$ the display name of the focused line
- $$SIPAUTHNAME$$ the SIP auth name of the focused line
- $$PROXYURL$$ the SIP proxy of the focused line
- $$INCOMINGNAME$$ returns the Caller-ID of the incoming call
- $$REMOTENUMBER$$ returns the number of the remote party (incoming or outgoing)

At the time of the HTTP call the variables are replaced with the value of the appropriate variable.

The variables are available in all the HTTP URL for programmable/soft keys as well as for the action URI. They can also be used in the XML objects themselves (TextMenu, InputScreen…).

Example

For example, if the administrator specifies an XML softkey with the value:

```
http://10.50.10.140/script.pl?name=$$SIPUSERNAME$$
```

This softkey executes a GET on:

```
http://10.50.10.140/script.pl?name=42512
```

assuming that the SIP username of the specific line is 42512.

The variables can be used, for instance, in a context where the phone has multiple SIP registrations and the XML application need to know which SIP registration the phone is currently using.

Note: the value of a variable depends on the status of the phone, if the variable has no meaning in the current status; the phone sends an empty string. For example if the URL is "http://myserver.com/script.pl?number=$$REMOTENUMBER$$&key=5", the phone will call "http://myserver.com/script.pl?number="&key=5" if the phone is idle.

The following table details when the variables are meaningful with an action URI.

	startup	registered	onhook	offhook	incoming	outgoing
$$SIPUSERNAME$$	X	X	X	X	X	X
$$DISPLAYNAME$$	X	X	X	X	X	X
$$SIPAUTHNAME$$	X	X	X	X	X	X
$$PROXYURL$$	X	X	X	X	X	X
$$INCOMINGNAME$$			X[2]		X	
$$REMOTENUMBER$$			X[2]		X	X

The following table details when the variables are meaningful for a programmable/soft key depending on the phone state.

	idle	connected	incoming	outgoing
$$SIPUSERNAME$$	X	X	X	X
$$DISPLAYNAME$$	X	X	X	X
$$SIPAUTHNAME$$	X	X	X	X
$$PROXYURL$$	X	X	X	X
$$INCOMINGNAME$$		X[3]	X	
$$REMOTENUMBER$$		X	X	X

[2] only if the previous call was an incoming call

[3] only if you have answered the current call.

5.3 XML Objects Pushed to the Phone

The phone can request an XML object via HTTP GET, or an object can be pushed to the phone via a POST. The phone parses this object immediately upon receipt and displays the information to the screen.

The HTTP POST packet must contain an "xml=" line in the message body. The string to parse is located after the equals sign in the message. HTML forms that post objects to the phone must use a field named "xml" to send their data. Any applications that construct HTTP packets on the fly must also specify this line.

Note: the content of the HTTP POST is not url-encoded.

Sample php source code

Below is a sample php source code that sends an XML object to an Aastra phone. In this example, the phone is located at 192.168.0.150 and the server pushing the XML object at 192.168.0.112.

```
<?php
#
function push2phone($server,$phone,$data)
{
$xml = "xml=".$data;

$post = "POST / HTTP/1.1\r\n";
$post .= "Host: $phone\r\n";
$post .= "Referer: $server\r\n";
$post .= "Connection: Keep-Alive\r\n";
$post .= "Content-Type: text/xml\r\n";
$post .= "Content-Length: ".strlen($xml)."\r\n\r\n";

$fp = @fsockopen ( $phone, 80, $errno, $errstr, 5);
if($fp)
      {
      fputs($fp, $post.$xml);
      flush();
      fclose($fp);
      }
}

##############################
$xml = "<AastraIPPhoneTextScreen>\n";
$xml .= "<Title>Push test</Title>\n";
$xml .= "<Text>This is a test for pushing a screen to a phone. It
is a way to demonstrate that we can push XML objects to an Aastra
Phone.</Text>\n";
$xml .= "</AastraIPPhoneTextScreen>\n";
push2phone("192.168.0.112","192.168.0.150",$xml);
?>
```

Sample perl source code

Below is a sample perl source code that sends an XML object to an Aastra phone. In this example, the phone is located at 192.168.0.150.

```
#!c:/perl/bin/Perl.exe
# Create a user agent object
     use LWP::UserAgent;
     use LWP::ConnCache;
     use HTTP::Request::Common;
my  $ua  =  LWP::UserAgent->new(agent  =>  'Mozilla/4.0  (compatible;
MSIE 5.5; Windows 98)');

$ua->conn_cache(LWP::ConnCache->new());
$ua->request(
   POST 'http://192.168.0.150'
        ,Content_Type  => 'application/x-www-form-urlencoded'
        ,Content       => 'xml=<AastraIPPhoneTextScreen><Title>Push
test</Title><Text>This is a test for pushing a screen to a phone.
It  is  a  way  to  demonstrate  that  we  can  push  XML  objects  to  an
Aastra Phone.</Text></AastraIPPhoneTextScreen>');
```

Notes:

- To accept a pushed page the “XML Push Server List” phone parameter must be configured with the list of the HTTP servers allowed pushing a page.
- When a page is pushed to the phone the MWI lamp blinks to indicate a new message to the phone.

When a XML object is pushed to the phone, the phone acts like a Web server listening to TCP port 80, which means that if the phone is behind a NAT and the server on the other side of the NAT, which is a typical configuration for a remote user, the port 80 must be forwarded to the IP phone at the router level.

6 Action URIs

Configuration parameters have been introduced to configure the phone to make an HTTP GET based on events, these events are:

- End of the boot sequence `action uri startup`
- Successful registration `action uri registered`
- On-hook `action uri onhook`
- Off-hook `action uri offhook`
- Incoming call `action uri incoming`
- Outgoing call `action uri outgoing`

6.1 Configuration

A URI can be configured for each type of events; the configuration can be made using:

- The configuration files
- The Web UI.

See chapter 7.4 for more details on the Web UI Configuration.

Configuration description

These URIs will be configurable via the configuration files using the following parameters.

```
action uri startup: <URI to GET on startup>
action uri registered: <URI to GET on successful registration>
action uri incoming: <URI to GET on an incoming call>
action uri outgoing: <URI to GET on outgoing call>
action uri offhook: <URI to GET on an off-hook event>
action uri onhook: <URI to get on an on-hook event>
```

As described in chapter 5.2 the action URI support variables in their configuration.

Example

```
action uri startup: http://myserver.com/startup.php
action uri incoming:
http://myserver.com/incoming.php?number=$$REMOTENUMBER$$
```

6.2 Action uri detailed behavior

Action uris are available on all the lines and line appearance of the phone so passing the system variables $$SIPUSERNAME$$ or $$SIPAUTHNAME$$ and $$PROXYURL$$ will define which line has triggered the action uri.

Action uri offhook

The offhook uri is triggered only when the user performs one of the following actions:

- Release the handset
- Press a line key

- Press the Speaker/Headset key

This means that the offhook uri is not triggered when the user presses a speeddial key even if the phone goes off-hook and dials a number. Also if the user directly dials a number and press the “dial” key the action uri is not triggered (but the outgoing uri is)

Action uri onhook

The onhook uri is triggered at the end of any active call:

- The user presses the Goodbye key
- The user hangs up the handset
- The user drops the call pressing the “drop” softkey
- After a completed transfer (blind or monitored)
- The other party hangs up.

Action uri incoming

The action uri incoming is triggered each time there is an incoming call presented to the phone and processed to make the phone ring. If the incoming call includes an auto-answer info message (intercom mode), the action uri incoming is triggered as well.

Note: the action uri incoming is not triggered if the phone is call forwarded or is in DND mode or if call waiting is disabled (and the phone in a non idle mode).

Action uri outgoing

The action uri outgoing is triggered each time an outgoing call is performed by the phone (SIP INVITE message sent).

Note: the action uri outgoing is also triggered when the user makes a second call to perform a transfer or a conference.

Action uri startup

The action uri startup is triggered at the end of the boot sequence.

Action uri register

The action uri register is triggered each time the phone registers successfully a line to its SIP proxy/registrar.

6.3 “Answer” and “Ignore” keys for action uri incoming

When a UI XML object is displayed as an answer to an action uri incoming, the regular contextual softkeys (“Answer” and “Ignore”) are overridden by the XML object softkeys, the user can only take the call by pressing the blinking line key or go off-hook.

To keep a consistent behavior of the phone after an action uri incoming, 2 custom softkeys have been introduced for the softkey phones (55i, 57i and 57i CT) as well as the “allowAnswer” tag for the non softkey phones (53i).

Softkey phones

The custom softkeys to use to display the “Answer” and “Ignore” button are “SoftKey:Answer” and “SoftKey:Ignore”.

The following softkeys declaration must be added to the UI XML object.

```
<SoftKey index="1">
<Label>Answer</Label>
<URI>SoftKey:Answer</URI>
</SoftKey>
<SoftKey index="2">
<Label>Ignore</Label>
<URI>SoftKey:Ignore</URI>
</SoftKey>
```

Once the call is answered or ignored, the 2 softkeys disappear automatically.

Non softkey phones

The “Answer” and “Ignore” will appear on the command line of the phone if the `allowAnswer` tag is added in the root tag of the UI XML object.

```
<AastraIPPhoneFormattedTextScreen allowAnswer="yes">
```

The 2 labels will disappear as soon as the call is answered or ignored.

6.4 Applications

Action URIs are very powerful as they allow an external application to take control of the display when an event occurs.

Here are some examples of potential applications.

Self-configuration

Using the startup URI, it is possible to develop self-provisioning on the phone. If a new phone boots and gets its configuration server IP address (through DHCP option 66 for example), it can download the `aastra.cfg` file with a startup URI set to an XML application, as it is a new phone the `<MAC>.cfg` config files does not exist. At the end of the boot, the phone will go to this XML application which can identify the phone and then generate the `<MAC>.cfg` file “on the fly” and ask the phone to reboot using the `PhoneExecute` object.

Screen pop

Using the action uri incoming, it is possible to display extra information on the phone for an incoming call. For instance, the XML application that is called when there is an incoming call can do a database lookup (Microsoft Exchange or any database) and display information on the caller. Basically it is like having a screen pop application directly on the phone.

Call Center

As for the Screen pop, the incoming URI can be used in a call center environment to display CRM or queue information on the caller. The on-hook URI can also be used to collect the wrap-up code at the end of a call.

And many more…

7 XML Configuration

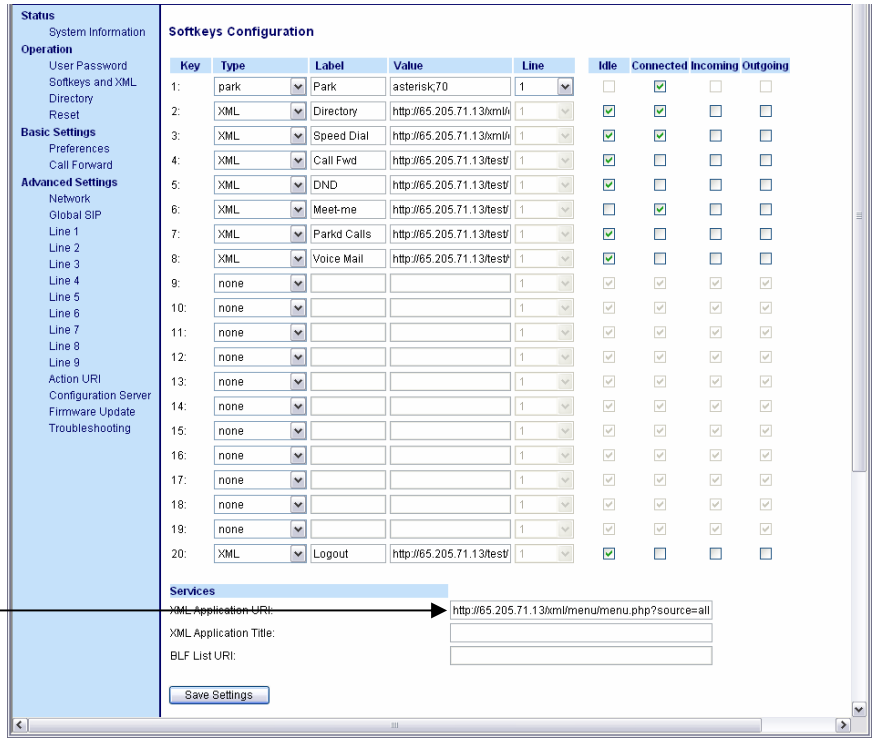
After creating an XML application to use on the IP phone, the application can be accessed as a Service or a Key.

7.1 Configuring a Custom Service from the Web UI (55i/57i/57i CT only)

To load a new custom XML application to the IP phone, enter an HTTP address for the application at the "**Softkeys and XML**" screen in the Aastra IP Phone Web interface.

1. Select Operation=>Softkeys and XML.
2. Enter the HTTP address in the "**XML Application URL**" field.
3. If desired, give the XML application a custom title in the "**XML Application Title**" field.

The following illustration shows an XML application named "*http://65.205.71.13/xml/menu/menu.php?source=all*" in the applicable field. After clicking Save Settings in the Softkeys Configuration screen, the XML application is dynamically applied to the IP phone you are configuring. The application URI can be accessed by pressing the "Services" button on your IP phone and selecting option 4.



XML Application URL

7.2 Configuring a Soft or Programmable Key from the Web UI

In addition to linking an XML application to a custom service, an application can be also be linked to a softkey and/or a programmable key depending on the phone and on the phone configuration (expansion modules).

1. Select "Operation" => "Softkeys and XML" (55i, 57i, 57iCT)

   Or

Select “Operation” => “Programmable Keys” (53i, 55i)

Or

Select “Operation” => “Expansion Module X” (53i, 55i, 57i, 57i CT with expansion module(s))

2. Choose type “XML” for the desired key.
3. Enter the URI in the value field.
4. Click Save Settings.

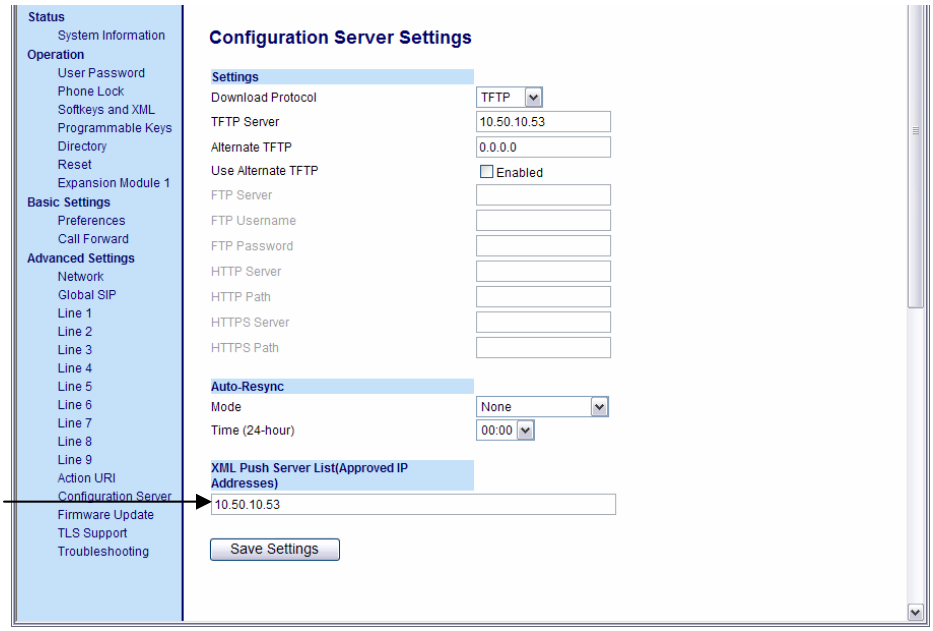
Note: Reboot is not necessary.

7.3 Configuring the XML Push Server List from the Web UI

The IP phone will only accept HTTP POSTs from the IP addresses set in the **XML Push Server List**.

1. Select Advanced Settings=>Configuration Server.
2. Enter a comma-separated list of IP addresses or domain names in the **XML Push Server List** field.
3. Click Save Settings and reboot the phone.

XML Push Server

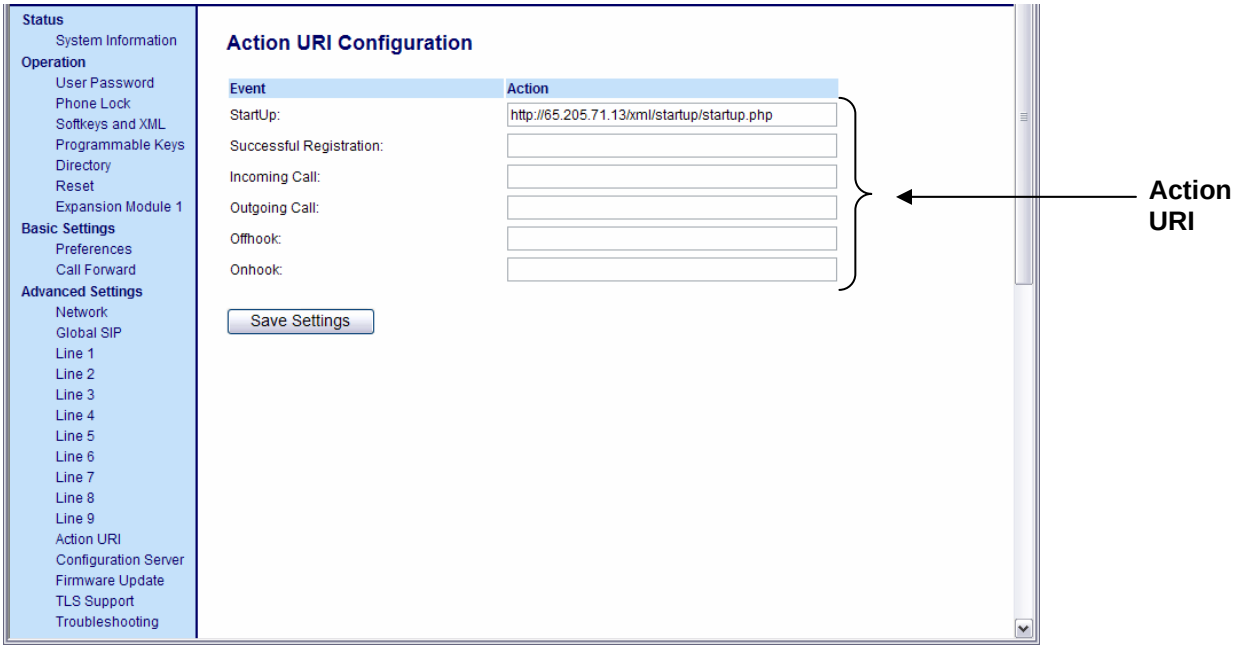


7.4 Configuring the Action URIs from the Web UI

You can configure the URI to be called for each type of event supported by the phone from the Web UI...

1. Select Advanced Settings=>Action URI.
2. Enter the URI for each type of event.

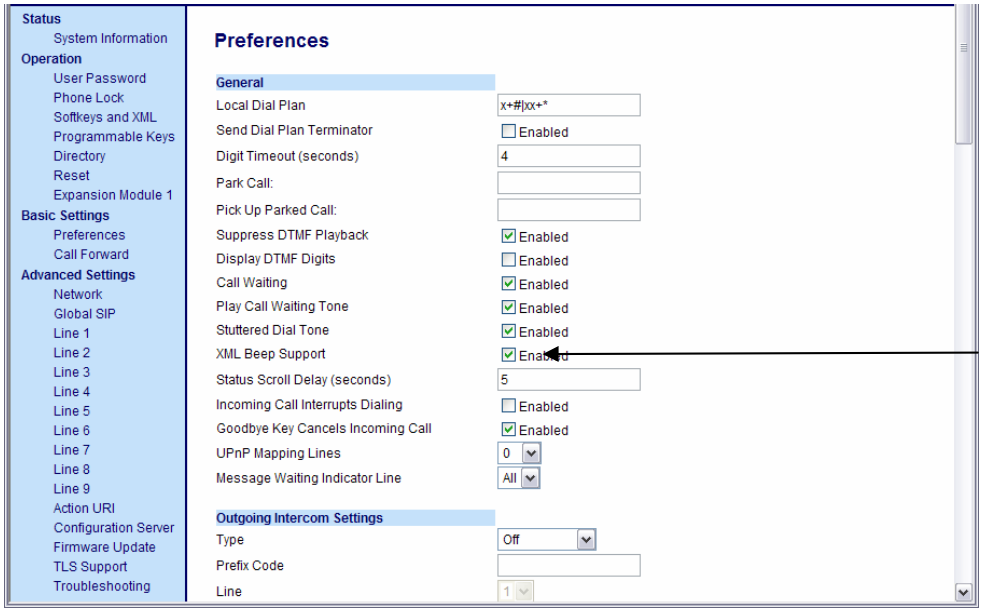
3. Click Save Settings.



7.5 Configuring the XML Beep Support from the Web UI

You can configure the **XML Beep Support** (enabled or disabled) from the Web UI. It impacts the behavior of the `AastraIPPhoneStatus` object regarding the phone notification.

1. Select Basic Settings=>Preferences.
2. Enable or disable the parameter.
3. Click Save Settings.

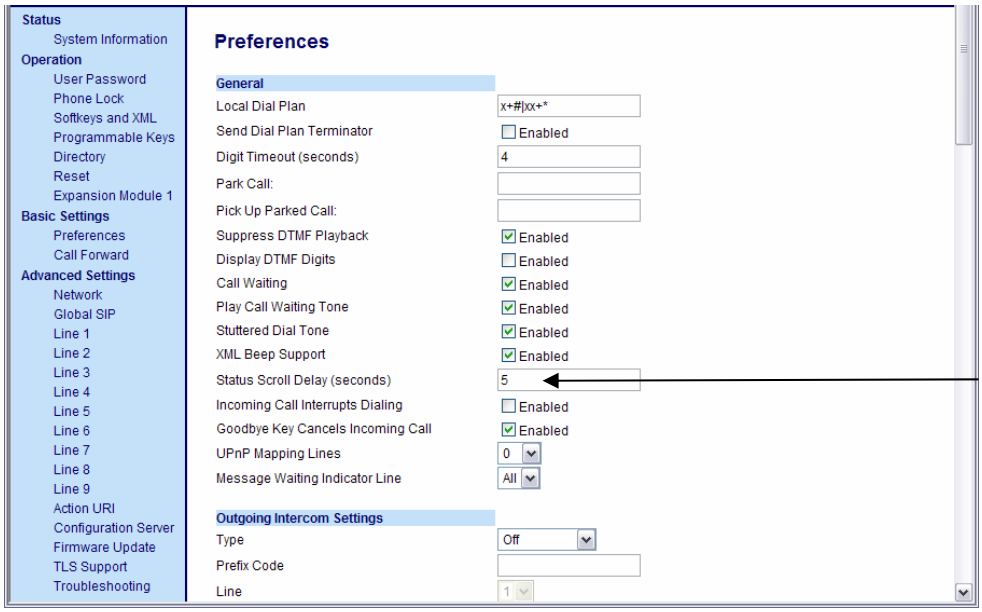


XML Beep Support

7.6 Configuring the Status Scroll Delay from the Web UI

You can configure the **Status Scroll Delay** (delay in seconds, default 5) from the Web UI. It impacts the behavior of the `AastraIPPhoneStatus` object defining the delay between each message.

1. Select Basic Settings=>Preferences.
2. Enter the value in seconds.
3. Click Save Settings.



Status Scroll Delay

7.7 XML Configuration using the Configuration Files

The *aastra.cfg* and *<mac>.cfg* Configuration File contains all the configuration parameters for the phone. Please refer to the phone administration guide for more details.

7.7.1 General XML parameters

You can enter an XML application in the *aastra.cfg* or *<mac>.cfg* file using the following parameters:

```
xml application URI
xml application title
xml application post list
xml beep notification
xml status scroll delay
xml get timeout
services script
callers list script
```

Parameter – `xml application URI`	Configuration Files aastra.cfg, <mac>.cfg
Description	This is the XML application you are loading into the IP phone configuration.
Format	HTTP server path or fully qualified Domain Name
Default Value	Not Applicable
Range	Not Applicable
Example	`xml application URI:` `http://172.16.96.63/aastra/internet.php`

Parameter – `xml application title`	Configuration Files aastra.cfg, <mac>.cfg
Description	This parameter allows you to rename the XML application in the IP phone UI (Services->4. Custom Feature). By default, when you load an XML application to the IP phone, the XML application title is called "Custom Feature". The "xml application title" parameter allows you to change that title. For example, if you are loading a traffic report XML application, you could change this parameter title to "Traffic Reports", and that title will display in the IP phone UI as Services->4. Traffic Reports.
Format	Alphanumeric characters
Default Value	Not Applicable
Range	Not Applicable
Example	`xml application title: Traffic Reports`

Parameter – `xml application post list`	Configuration Files aastra.cfg, <mac>.cfg
Description	The HTTP server that is pushing XML applications to the IP phone.
Format	IP address in dotted decimal format and/or Domain name address
Default Value	Not Applicable
Range	Not Applicable
Example	`xml application post list: 10.50.10.53,` `dhcp10-53.ana.aastra.com`

Parameter – `xml beep notification`	Configuration Files aastra.cfg, <mac>.cfg
Description	Enables or disables a BEEP notification on the phone when an AastraIPPhoneStatus object containing a “beep” attribute arrives to the phone.
Format	Boolean
Default Value	Value 1 (ON)
Range	0 (OFF) No beep is audible even if the beep attribute is present in the XML object. 1 (ON) The phone beeps when an XML object with the “beep” attribute arrives to the phone.
Example	`xml beep notification: 0`

Parameter – `xml status scroll delay`	Configuration Files aastra.cfg, <mac>.cfg
Description	Specifies the length of time, in seconds, that each XML status message displays on the phone.
Format	Integer
Default Value	5
Range	1 to 25
Example	`xml status scroll delay: 3`

Parameter – `xml get timeout`	Configuration Files aastra.cfg, <mac>.cfg
Description	Specifies the length of time, in seconds, that the phone will wait for a HTTP GET answer called by an XML.
Format	Integer
Default Value	0 (no timeout)
Range	0 to 4294967295
Example	`xml get timeout: 10`

Parameter – `services script`	Configuration Files aastra.cfg, <mac>.cfg
Description	Specifies the XML URI to call when a “Services” key (prgkey or softkey) is pressed. The URI overrides native behavior of the “Services” key.
Format	HTTP server path or fully qualified Domain Name
Default Value	Not Applicable
Range	Not Applicable
Example	`services script: http://172.16.96.63/services.php`

Parameter – `callers list script`	Configuration Files aastra.cfg, <mac>.cfg
Description	Specifies the XML URI to call when a “Callers List” key (prgkey or softkey) is pressed. The URI overrides native behavior of the “Callers List” key.
Format	HTTP server path or fully qualified Domain Name
Default Value	Not Applicable
Range	Not Applicable
Example	`callers list script: http://172.16.96.63/callers.php`

7.7.2 Programmable and Soft keys

You can configure keys calling an XML application using the following parameters.

```
softkeyX/prgkeyX/topsoftkeyX/expmodT keyX type: xml
softkeyX/prgkeyX/topsoftkeyX/expmodT keyX value: http://someapp.xml
```

As described in chapter 5.2, system variables can be used in the URI to be called by pressing a key.

7.7.3 Examples

Example (53i)

```
# XML configuration
xml application URI: http://172.16.96.63/aastra/internet.php
xml application post list: 10.10.50.53, xmlserver.aastra.com
xml beep notification: 1
xml status scroll delay: 5

# Softkey 1
softkey1 type: xml
softkey1 label: My XML
softkey1 value: http://172.16.96.63/aastra/internet.php

# Softkey 2
softkey2 type: xml
softkey2 label: Login
softkey2 value: http://myserver.com/login.php?user=$$SIPUSERNAME$$
```

Example (55i/57i/57i CT)

```
# XML configuration
xml application URI: http://172.16.96.63/aastra/internet.php
xml application post list: 10.10.50.53, xmlserver.aastra.com
xml beep notification: 1
xml status scroll delay: 5

# Key 1
```

```
prgkey1 type: xml
prgkey1 value: http://172.16.96.63/aastra/internet.php

# Key 2
prgkey2 type: xml
prgkey2 value: http://myserver.com/login.php?user=$$SIPUSERNAME$$
```

8 Why XML Applications for an IP Phone?

8.1 Telephony applications

This chapter details potential XML telephony applications which could be developed to enhance integration of an IP phone with the other telecom applications.

8.1.1 Directory

The first obvious applications that can be developed are the directory application, it includes:

- PBX directory
- Corporate directory (Global list from a Microsoft Exchange server)
- Personal contacts (My contacts in Outlook)
- Any LDAP directory (public or private)

8.1.2 Call Processing

XML applications can also be used to develop interactions between the call processing and the phone:

- DND
- Call Forward
- Parked calls
- Call pickup
- PBX configuration

8.1.3 Voice-Mail

- Voice mail messages management (play, skip, delete, …)
- Display message envelopes
- Presence management

8.1.4 Conference bridge

- Conference booking
- Conference reminder
- Audio console

8.1.5 Contact Center

- Agent Login/Logout

- Access to call center reports
- Account codes
- Wrap codes

8.2 Vertical applications

This chapter details potential applications that can be developed as an XML application for the Aastra IP phones. The list is far from being exhaustive.

8.2.1 Human Resources

- Available Vacation Days
- Available Personal Days
- 401K balance
- Clock-In/Clock-Out

8.2.2 Travel/Hotel

- Current Balance
- In-Room Dining Ordering
- Delivery Dining Options
- Extend Stay
- Schedule Airport Shuttle
- Request Housekeeping/Engineering
- Leave Feedback
- Wake-Up Call
- Book Corporate Travel
- Do Not Disturb

8.2.3 General Mobile Phone

- Weather Alerts
- Stock Alerts
- Stock Prices
- Worldwide Time/Temperature
- Server Alarms and Notifications

- Server Status
- Account Balances
- Current Gas Prices
- Local Movie Times
- Upcoming Concerts-By Category
- Order Flowers-by Category
- Send Order to Starbucks
- Send SMS Messages

8.2.4 Health Care

- Test Results
- Manage Appointment
- Appointment Reminders
- Take-Your-Medicine Reminders
- Order Hospital Meals
- Check Pharmacy Inventory
- Schedule Blood Donation

8.2.5 Education

- Attendance
- Request Substitute Teacher (used by primary teacher)
- Review Open Requests for Substitute Teacher (used by potential subs)
- Schedule Classes
- Request Dorm Room Change
- School Closing Notification/Status
- Parent Contact Info

8.2.6 General

- Track FedEx Package (or Airborne, UPS, etc.)
- Calling Card Minutes Remaining
- Reserve Meeting Rooms

- Contact center Metrics (Calls Waiting, Longest Hold, Performance against Service Level, etc.)
- Pro Sports Scores, Vegas Betting Lines
- Multitude of Banking apps: Balances, Transfers, etc.
- Music/Marketing On Hold Options
- Language Translation
- Daily Horoscope
- Broadcast Joke Of The Day/Inspirational Quote of the Day

8.2.7 Law Enforcement

- Amber Alerts
- Traffic Ticket Plead By Phone
- Fugitive Alerts

9 Phone Self-Configuration using XML

9.1 Introduction

The deployment of a SIP phone is not a simple task; you have to face 2 challenges:

1. Provide the address of the configuration server to the phone
2. Link the MAC address of the phone with a SIP extension

The first challenge is usually solved by using DHCP option 66 (bootp) to provision the phone with the IP address/name of the configuration server (TFTP, FTP, HTTP or HTTPS).

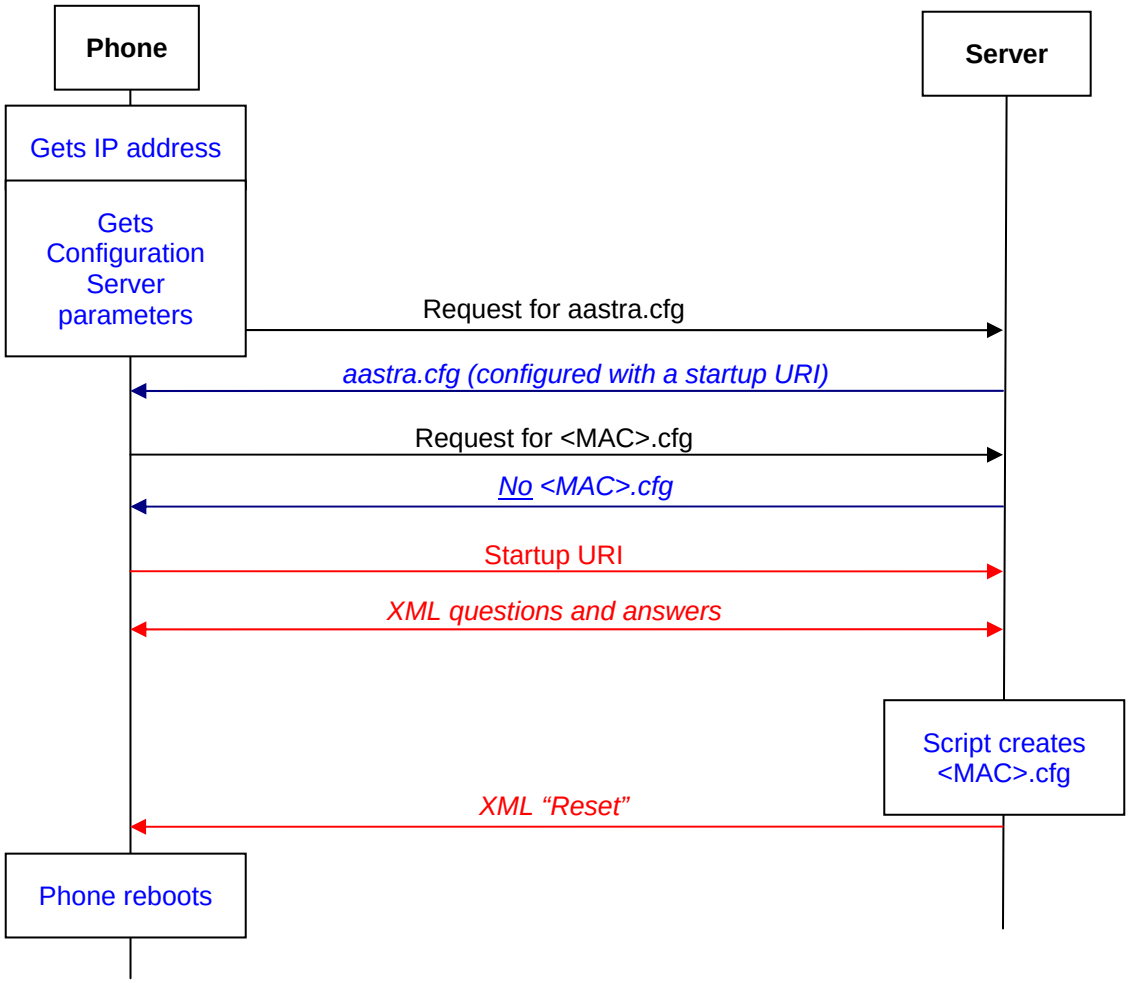
The second challenge is more difficult as you have to know the MAC address of the phone in advance in order to prepare the specific configuration file the phone will use. Usually, each phone is identified (scan of the MAC address) and then linked to an extension.

It is possible to have a complete self-configuration of the phone using an XML application called by the action uri startup at the end of the boot sequence.

9.2 Message flow

It is possible for a third-party to develop an automatic configuration process. The following is a description of how this can be done using existing phone features.

1. The aastra.cfg file sets the startup action uri configuration parameter to point to the configuration script and configuration download information.
2. Phone downloads the aastra.cfg file, ignores missing <MAC>.cfg file and continues boot process.
3. Phone executes startup uri, running the configuration script. The MAC address of the phone and the phone model are in the HTTP headers of the request.
4. The script uses XML to gather required configuration information and creates <mac>.cfg file. The <mac>.cfg file must reset the startup action uri to avoid the configuration script being called on subsequent boots.
5. The script reboots the phone via XML reboot command or via SIP check-sync message.
6. Phone reboots, directly downloads both aastra.cfg and newly created <MAC>.cfg file



9.3 Auto-configuration policy

The auto-configuration policy which includes the flow of questions asked to the user and the script to generate the <MAC>.cfg file is totally open with this mechanism.

Multiple options are available.

Extension is already provisioned in the IP-PBX database.

One way to implement this feature might be to have all the extensions already provisioned on the switch side and the XML flow will be used to identify the user (extension number and voicemail password for instance). The script must then control if the extension is not already assigned and create the <MAC>.cfg based on the extension(s) configuration.

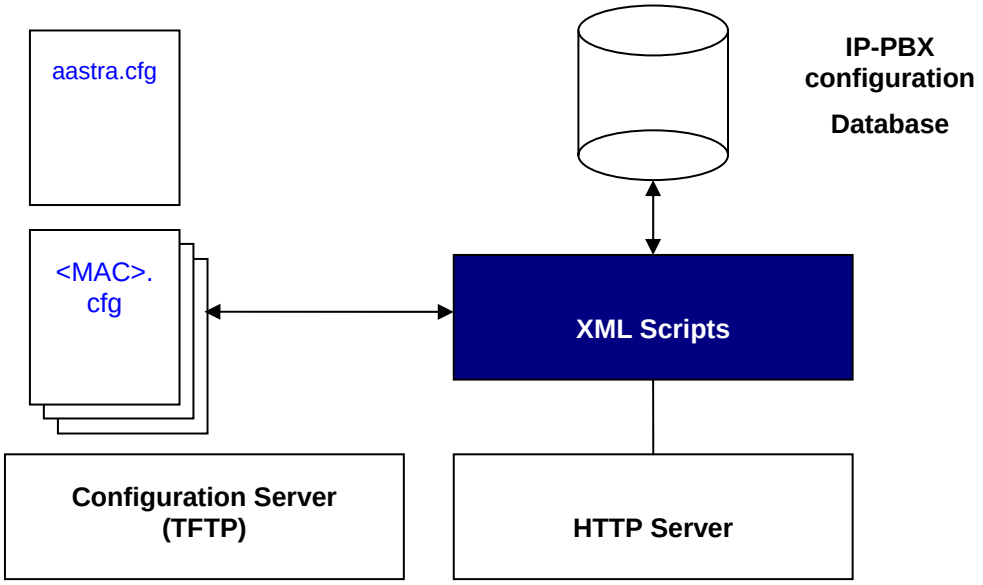
Another way would be to display the list of available extensions and let the user select his extension; of course password protection is needed to avoid any hacking of the platform.

Extension is not provisioned in the IP-PBX database

A second option is to have the script to provision the extension in the database. To do so, the script can ask for user general parameters (name…) and automatically creates an extension in the switch database and then creates the <MAC>.cfg, the extension number can be either automatically assigned or the user can select it in a list of available extensions.

9.4 Architecture

The following diagram represents the architecture of what needs to be developed to implement the Aastra self-configuration mechanism.



The development effort to implement the self-configuration is fairly straight forward but the complexity depends on the policy to attribute extensions on the phone system.

9.5 Sample implementation

Aastra provides here a possible implementation for the self-configuration, this is just an example source code not supported by Aastra Telecom.

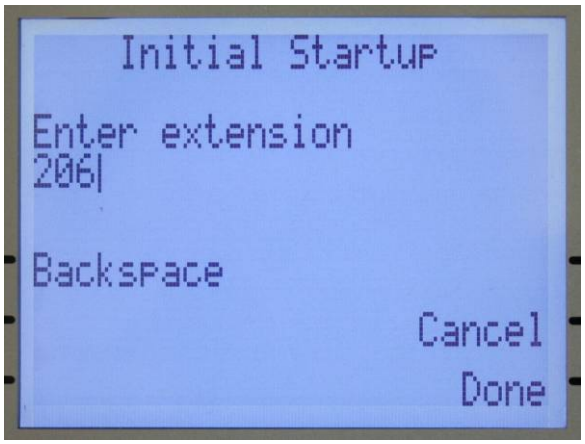
In this implementation targeted for Asterisk, the chosen policy is to have all the extensions pre-configured in the Asterisk database and use

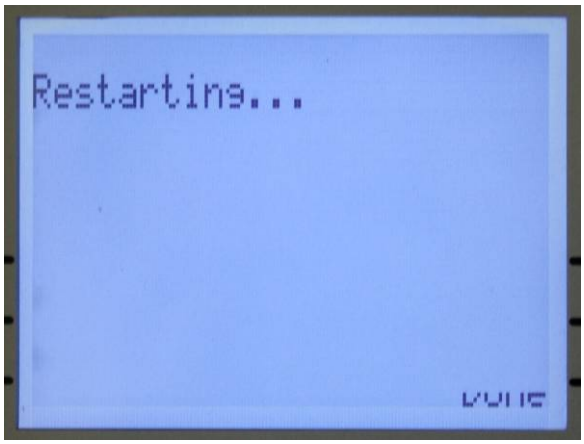
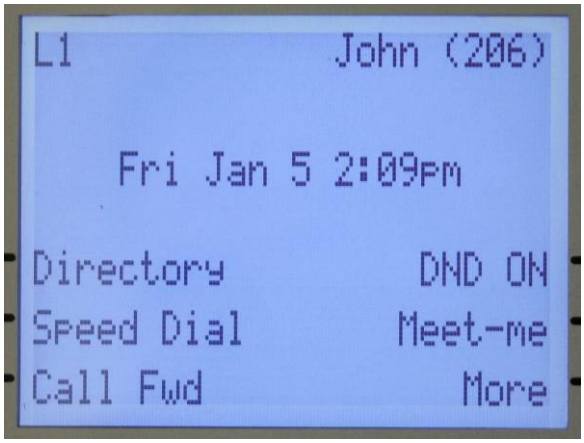
- the extension
- the voice mail password

for the credentials to authenticate the user and link the MAC address of the phone to an extension.

See chapter 14 for the source code.

9.6 Screenshots

	End of the boot sequence aastra.cfg downloaded <MAC>.cfg not downloaded
	Action uri startup has been called XML application asks for the extension
	XML application asks for the VM password

	Phone restarts
	Phone is ready to use.

10 Free XML applications available on the Web

Aastra Telecom has made available, **for demonstration purpose only**, a list of XML applications on the Internet.

The applications currently available are:

- Area code
- Ask Google
- CNN News
- ESPN News
- Horoscope
- Movies
- Stock
- Today…
- Weather
- World Clock

Note: Aastra Telecom does not guarantee the availability of these applications. The applications can change any time without notice.

Notes:

- Aastra Telecom does not guarantee the availability of these applications. The applications can change any time without notice.
- These applications should not be used commercially; any abusive use of these applications will be detected and the phone will be automatically banned from the applications.
- your Aastra SIP phone must have Internet access to use these applications.

These XML applications can be configured individually or as a global menu.

10.1 Area code

This application allows the user to lookup for the State/Cities of any given area code.

```
uri=http://65.205.71.13/xml/area/area.php
```

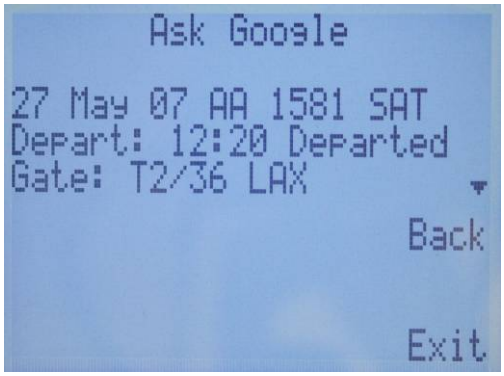
10.2 Ask Google

This application allows the user to ask a general question to Google.com. Please refer to http://www.google.com/sms for more details.

Questions

- 1 USD in Euro
- Define TCP
- Translate bread in French
- …

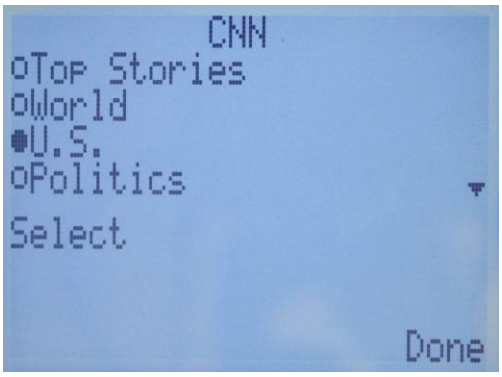
```
uri="http://65.205.71.13/xml/google/google.php"
```

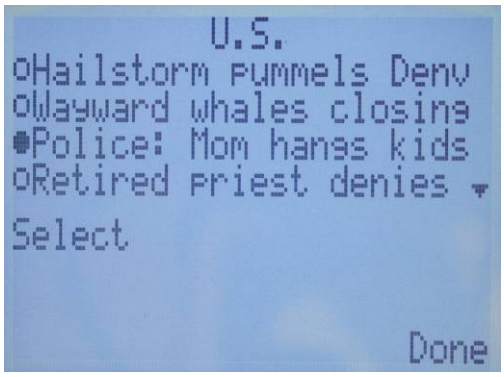


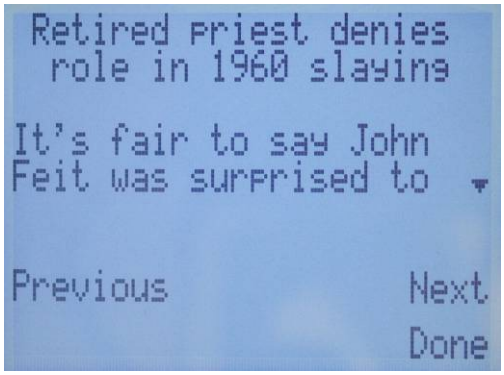
10.3 CNN News

RSS feed from CNN.com including Top Stories, World, Politics…

```
uri="http://65.205.71.13/xml/rss/rss.php?feed=cnn"
```







10.4 ESPN News

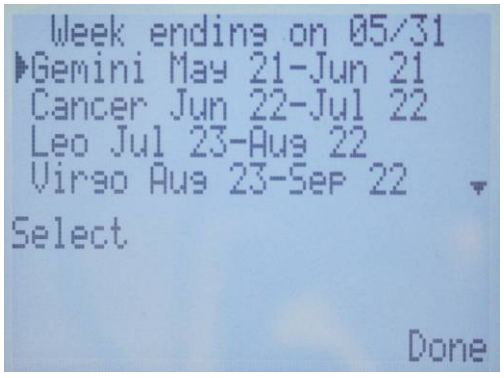
RSS feed from ESPN.com bringing news for the most popular sports in North America.

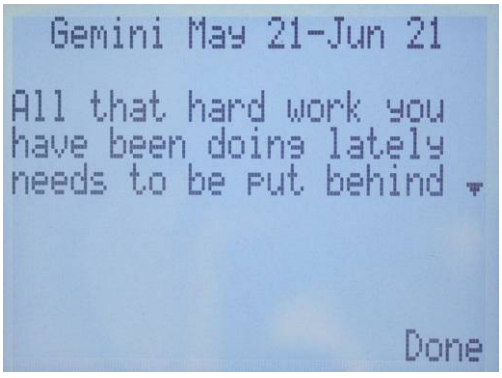
```
uri="http://65.205.71.13/xml/rss/rss.php?feed=espn"
```

10.5 Horoscope

RSS feed from dailyhoroscopes.com.

```
uri="http://65.205.71.13/xml/horoscope/horoscope.php"
```





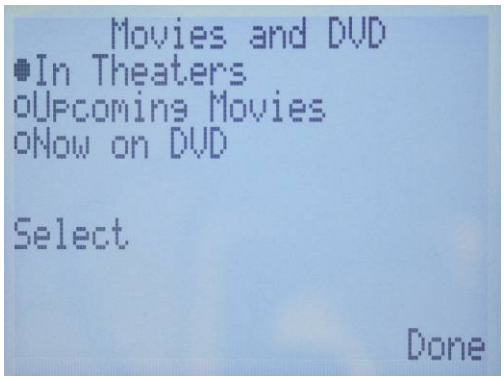
10.6 Movies

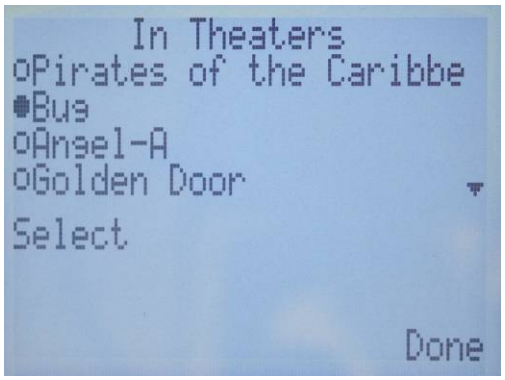
RSS feed from movies.com including the following topics:

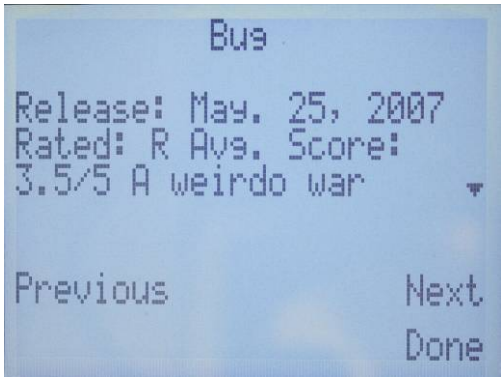
- In theaters

- Upcoming movies
- Now on DVD

```
uri="http://65.205.71.13/xml/rss/rss.php?feed=movies"
```







10.7 Stock

This application uses www.yahoo.com to get the value of any given stock. Please refer to yahoo.com for the syntax of the stock ticker.

```
uri="http://65.205.71.13/xml/stock/stock.php"
```

10.8 Today…

RSS feed from answers.com including the following topics:

- Word of the Day
- Birthdays today
- This day in History
- Quote of the Day

```
uri="http://65.205.71.13/xml/rss/rss.php?feed=day"
```

10.9 Weather

RSS feed from rssweather.com providing weather forecast for the following cities:

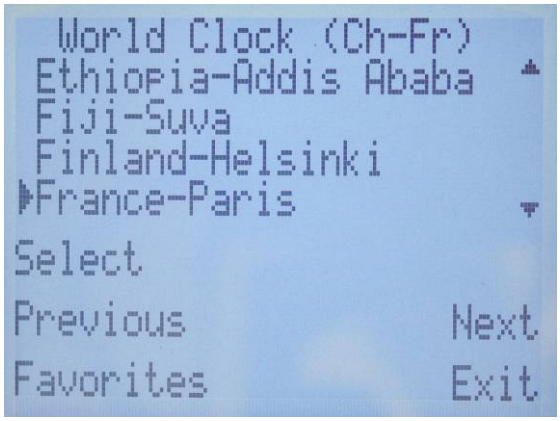
- Boston, MA, USA
- Toronto, Canada
- Billerica, MA, USA
- Montreal, Canada
- Frisco, TX, USA
- Berlin, Germany
- Paris, France
- Zurich, Switzerland

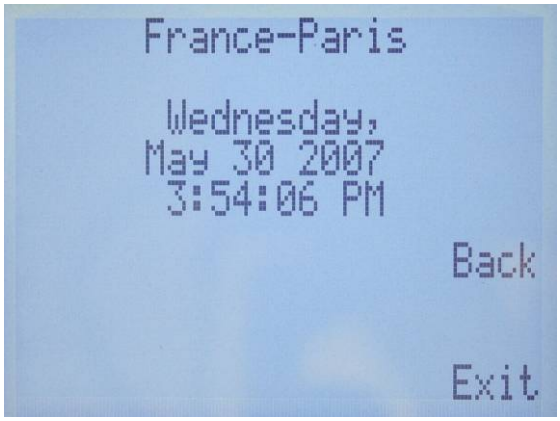
```
uri="http://65.205.71.13/xml/rss/rss.php?feed=weather
```

10.10 World Clock

Date and time from around the world using www.timeanddate.com

```
uri="http://65.205.71.13/xml/clock/clock.php"
```

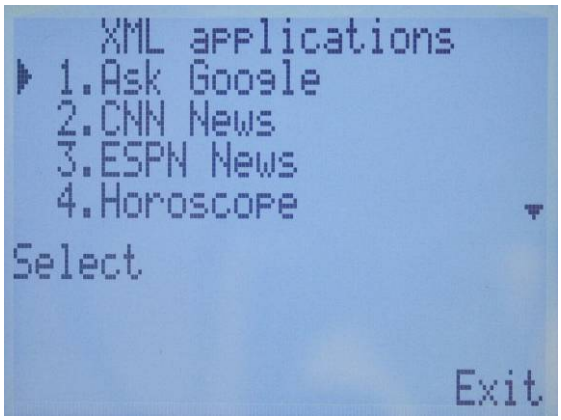




10.11 Global menu

The following URI displays a menu with all the above applications. It can be used as the “Custom feature” menu on a SIP Phone.

```
uri=”http://65.205.71.13/xml/menu/menu.php?source=all”
```



11 Appendix A: XSL Model

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:simpleType name="textAttributeType">
 <xs:restriction base="xs:string">
  <xs:pattern value="yes|no" />
 </xs:restriction>
</xs:simpleType>

<xs:simpleType name="integerAttributeType">
 <xs:restriction base="xs:integer" />
</xs:simpleType>

<xs:simpleType name="verticalAlignType">
 <xs:restriction base="xs:string">
  <xs:pattern value="top|middle|bottom" />
 </xs:restriction>
</xs:simpleType>

<xs:simpleType name="horizontalAlignType">
 <xs:restriction base="xs:string">
  <xs:pattern value="left|middle|right" />
 </xs:restriction>
</xs:simpleType>

<xs:complexType name="titleTagType" mixed="true">
  <xs:attribute name="wrap" default="yes">
     <xs:simpleType>
       <xs:restriction base="xs:string">
         <xs:pattern value="yes|no"/>
       </xs:restriction>
     </xs:simpleType>
  </xs:attribute>
</xs:complexType>

<xs:complexType name="lineTagType" mixed="true">
 <xs:attribute name="Size" default="single">
  <xs:simpleType>
   <xs:restriction base="xs:string">
    <xs:pattern value="double|single" />
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
 <xs:attribute name="Align" default="left">
  <xs:simpleType>
   <xs:restriction base="xs:string">
    <xs:pattern value="right|left|center" />
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
</xs:complexType>

<xs:complexType name="imageType"        mixed="true">
 <xs:attribute  name="verticalAlign"    type="verticalAlignType"    />
 <xs:attribute  name="horizontalAlign"  type="horizontalAlignType"  />
 <xs:attribute name="height">
  <xs:simpleType>
   <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"  />
    <xs:maxInclusive value="40" />
```

```
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
 <xs:attribute  name="width" type="integerAttributeType" default="0" />
</xs:complexType>

<xs:complexType name="softKeyType">
 <xs:sequence>
  <xs:element name="Label" type="xs:string" />
  <xs:element name="URI" type="xs:string" />
 </xs:sequence>
 <xs:attribute name="index" use="required">
  <xs:simpleType>
   <xs:restriction base="xs:integer">
    <xs:minInclusive value="1" />
    <xs:maxInclusive value="6" />
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
 <xs:attribute name="icon" type="integerAttributeType" default="0" />
</xs:complexType>

<xs:complexType name="iconListType">
 <xs:sequence>
  <xs:element name="Icon" minOccurs="1" maxOccurs="unbounded">
   <xs:complexType mixed="true">
    <xs:attribute name="index" type="integerAttributeType" use="required"
/>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
</xs:complexType>

<xs:element name="AastraIPPhoneTextScreen">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Title" type="titleTagType" minOccurs="0"
maxOccurs="1"/>
   <xs:element name="Text">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="1000" />
     </xs:restriction>
    </xs:simpleType>
   </xs:element>
   <xs:element  name="SoftKey" type="softKeyType" minOccurs="0"
maxOccurs="6"/>
  </xs:sequence>
  <xs:attribute name="destroyOnExit" type="textAttributeType"
default="no" />
  <xs:attribute name="Beep"          type="textAttributeType"
default="no" />
  <xs:attribute name="LockIn"        type="textAttributeType"
default="no" />
  <xs:attribute name="Timeout"       type="integerAttributeType"
default="45"  />
 <xs:attribute name="cancelAction"  type="xs:string" />
 <xs:attribute name="allowAnswer" type="xs:string" default="no"/>
 </xs:complexType>
</xs:element>

<xs:element name="AastraIPPhoneTextMenu">
```

```
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Title" type="titleTagType" minOccurs="0"
maxOccurs="1"/>
   <xs:element name="MenuItem" minOccurs="1" maxOccurs="15">
    <xs:complexType>
     <xs:all>
      <xs:element name="Prompt" type="xs:string" minOccurs="0"
maxOccurs="1" />
      <xs:element name="URI" type="xs:string" minOccurs="0" maxOccurs="1"
/>
      <xs:element name="Dial" type="xs:string" minOccurs="0" maxOccurs="1"
/>
      <xs:element name="Selection" type="xs:string" minOccurs="0"
maxOccurs="1" />
     </xs:all>
     <xs:attribute name="base" type="xs:string" />
     <xs:attribute name="icon" type="integerAttributeType"  default="0" />
    </xs:complexType>
   </xs:element>
   <xs:element name="SoftKey"  type="softKeyType"  minOccurs="0"
maxOccurs="6" />
   <xs:element name="IconList" type="iconListType" minOccurs="0"
maxOccurs="1" />
  </xs:sequence>
  <xs:attribute name="destroyOnExit" type="textAttributeType"
default="no" />
  <xs:attribute name="Beep"          type="textAttributeType"
default="no" />
  <xs:attribute name="LockIn"        type="textAttributeType"
default="no" />
  <xs:attribute name="Timeout"       type="integerAttributeType"
default="45"  />
  <xs:attribute name="defaultIndex"  type="integerAttributeType"
default="1"  />
  <xs:attribute name="cancelAction"  type="xs:string" />
  <xs:attribute name="style" type="xs:string" default="numbered" />
  <xs:attribute name="allowAnswer" type="xs:string" default="no"/>
 </xs:complexType>
</xs:element>

<xs:element name="AastraIPPhoneInputScreen">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Title" type="titleTagType" minOccurs="0"
maxOccurs="1" />
   <xs:element name="Prompt" minOccurs="0" maxOccurs="1" />
   <xs:element name="URL" />
   <xs:element name="Parameter" minOccurs="0" maxOccurs="1" />
   <xs:element name="Default" minOccurs="0" maxOccurs="1" />
   <xs:element name="Selection" minOccurs="0" maxOccurs="1" />
   <xs:element name="InputField" minOccurs="0" maxOccurs="6">
      <xs:complexType>
         <xs:sequence>
           <xs:element name="Prompt" minOccurs="0" maxOccurs="1" />
           <xs:element name="Parameter" minOccurs="0" maxOccurs="1" />
           <xs:element name="Default" minOccurs="0" maxOccurs="1" />
            <xs:element name="Selection" minOccurs="0" maxOccurs="1" />
           <xs:element name="SoftKey" type="softKeyType" minOccurs="0"
maxOccurs="6"/>
         </xs:sequence>
         <xs:attribute name="type" use="optional">
            <xs:simpleType>
```

```
              <xs:restriction base="xs:string">
                <xs:pattern
value="IP|string|number|timeUS|dateUS|timeInt|dateInt|Empty" />
             </xs:restriction>
          </xs:simpleType>
       </xs:attribute>
       <xs:attribute name="password"      type="textAttributeType"
default="no"  />
       <xs:attribute name="editable"      type="textAttributeType"
default="yes" />
     </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="type" use="required">
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:pattern value="IP|string|number|timeUS|dateUS|timeInt|dateInt" />
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="password"      type="textAttributeType"
default="no"  />
  <xs:attribute name="destroyOnExit" type="textAttributeType"
default="no"  />
  <xs:attribute name="editable"      type="textAttributeType"
default="yes" />
  <xs:attribute name="Beep"          type="textAttributeType"
default="no"  />
  <xs:attribute name="LockIn"        type="textAttributeType"
default="no" />
  <xs:attribute name="Timeout"       type="integerAttributeType"
default="45" />
  <xs:attribute name="defaultIndex"  type="integerAttributeType"
default="1" />
  <xs:attribute name="cancelAction"  type="xs:string" />
  <xs:attribute name="allowAnswer" type="xs:string" default="no"/>
  <xs:attribute name="displayMode" default="uncondensed">
    <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:pattern value="condensed|uncondensed" />
       </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
  </xs:complexType>
</xs:element>

<xs:element name="AastraIPPhoneDirectory">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Title" type="xs:string" />
   <xs:element name="MenuItem" minOccurs="1" maxOccurs="15">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="Prompt" type="xs:string" />
      <xs:element name="URI" type="xs:string" />
     </xs:sequence>
     <xs:attribute name="base" type="xs:string" />
     <xs:attribute name="icon" type="integerAttributeType"  default="0" />
    </xs:complexType>
   </xs:element>
   <xs:element name="SoftKey" type="softKeyType" minOccurs="0"
maxOccurs="6"   />
```

```
   <xs:element name="IconList" type="iconListType" minOccurs="0"
maxOccurs="1" />
  </xs:sequence>
  <xs:attribute name="destroyOnExit" type="textAttributeType"
default="no" />
  <xs:attribute name="Beep"          type="textAttributeType"
default="no" />
  <xs:attribute name="LockIn"        type="textAttributeType"
default="no" />
  <xs:attribute name="Timeout"       type="integerAttributeType"
default="45" />
  <xs:attribute name="next"          type="xs:string" />
  <xs:attribute name="previous"      type="xs:string" />
   <xs:attribute name="cancelAction"  type="xs:string" />
 </xs:complexType>
</xs:element>

<xs:element name="AastraIPPhoneExecute">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="ExecuteItem" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:attribute name="URI" type="xs:string" />
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="Beep" type="textAttributeType" default="no" />
  <xs:attribute name="triggerDestroyOnExit"  type="textAttributeType"
default="no"  />
 </xs:complexType>
</xs:element>

<xs:element name="AastraIPPhoneStatus">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Session" type="xs:string" minOccurs="0" />
   <xs:element name="Message">
    <xs:complexType mixed="true">
     <xs:attribute name="index"   type="integerAttributeType"
use="required" />
     <xs:attribute name="type">
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:pattern value="alert" />
       </xs:restriction>
      </xs:simpleType>
     </xs:attribute>
     <xs:attribute name="Timeout" type="integerAttributeType" default="3"
/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="Beep" type="textAttributeType" default="no" />
  <xs:attribute name="triggerDestroyOnExit"  type="textAttributeType"
default="no"  />
 </xs:complexType>
</xs:element>

<xs:element name="AastraIPPhoneConfiguration">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="ConfigurationItem" minOccurs="0"
maxOccurs="unbounded">
```

```
    <xs:complexType>
     <xs:sequence>
      <xs:element name="Parameter" type="xs:string"  />
      <xs:element name="Value"     type="xs:string"  />
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="Beep" type="textAttributeType"  default="no"  />
  <xs:attribute name="triggerDestroyOnExit"  type="textAttributeType"
default="no"  />
 </xs:complexType>
</xs:element>

<xs:group name="linesAndScroll">
   <xs:sequence>
       <xs:element name="Scroll" minOccurs="1" maxOccurs="1">
              <xs:complexType>
            <xs:sequence>
               <xs:element name="Line" type="lineTagType" minOccurs="1"
maxOccurs="unbounded" />
             </xs:sequence>
             <xs:attribute name="Height" type="integerAttributeType"
default="1" />
          </xs:complexType>
       </xs:element>
       <xs:element name="Line" type="lineTagType" minOccurs="0"
maxOccurs="unbounded" />
  </xs:sequence>
</xs:group>

<xs:element name="AastraIPPhoneFormattedTextScreen">
 <xs:complexType>
  <xs:sequence>
    <xs:element name="Line" type="lineTagType" minOccurs="0"
maxOccurs="unbounded" />
    <xs:group ref="linesAndScroll" minOccurs="0" maxOccurs="1"/>
    <xs:element  name="SoftKey" type="softKeyType" minOccurs="0"
maxOccurs="6"    />
  </xs:sequence>
  <xs:attribute name="destroyOnExit"  type="textAttributeType"
default="no"  />
  <xs:attribute name="Beep"           type="textAttributeType"
default="no"  />
  <xs:attribute name="LockIn"        type="textAttributeType"
default="no" />
  <xs:attribute name="Timeout"       type="integerAttributeType"
default="45"  />
  <xs:attribute name="cancelAction"  type="xs:string" />
  <xs:attribute name="allowAnswer" type="xs:string" default="no"/>
 </xs:complexType>
</xs:element>

<xs:element name="AastraIPPhoneImageScreen">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Image" type="imageType" />
   <xs:element name="SoftKey" type="softKeyType" minOccurs="0"
maxOccurs="6" />
  </xs:sequence>
  <xs:attribute  name="destroyOnExit" type="textAttributeType"
default="no"  />
```

```
  <xs:attribute  name="Beep"          type="textAttributeType"
default="no"  />
  <xs:attribute  name="LockIn"        type="textAttributeType"
default="no" />
  <xs:attribute  name="Timeout"       type="integerAttributeType"
default="45" />
  <xs:attribute name="cancelAction"  type="xs:string" />
 </xs:complexType>
</xs:element>

<xs:element name="AastraIPPhoneImageMenu">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Image" type="imageType" />
   <xs:element name="URIList">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="URI" minOccurs="0" maxOccurs="12">
       <xs:complexType mixed="true">
        <xs:attribute name="base" />
        <xs:attribute name="key" use="required" >
         <xs:simpleType>
          <xs:restriction base="xs:string">
           <xs:pattern value="[0-9]|#|\*" />
          </xs:restriction>
         </xs:simpleType>
        </xs:attribute>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="SoftKey"   type="softKeyType"  minOccurs="0"
maxOccurs="6" />
   <xs:element name="IconList"  type="iconListType" minOccurs="0"
maxOccurs="1" />
  </xs:sequence>
  <xs:attribute name="destroyOnExit" type="textAttributeType"
default="no"  />
  <xs:attribute name="Beep"          type="textAttributeType"
default="no"  />
  <xs:attribute name="LockIn"        type="textAttributeType"
default="no"  />
  <xs:attribute name="Timeout"       type="integerAttributeType"
default="45"   />
  <xs:attribute name="cancelAction"  type="xs:string" />
  <xs:attribute name="allowAnswer" type="xs:string" default="no"/>
 </xs:complexType>
</xs:element>

</xs:schema>
```

12 Appendix B: Object Oriented PHP Classes

Aastra Telecom provides an object oriented API to develop XML applications. The API is available on the Aastra Web site http://www.aastratelecom.com.

Note: The PHP objects are taking care of the XML escape encoding when they are needed.

12.1 AastraIPPhoneConfiguration()

This class allows you to create a XML PhoneConfiguration object.

Include

AastraIPPhoneConfiguration.class.php

Methods

- output() to display the object
- setBeep() to enable a notification beep with the object (optional)
- addEntry(parameter,value) to add a configuration change.
- setTriggerDestroyOnExit() to set the triggerDestroyOnExit tag to "yes" (optional)
- generate() to return the object content as a string for debug purpose

Example

```
require_once('AastraIPPhoneConfiguration.class.php');
$configuration = new AastraIPPhoneConfiguration();
$configuration->addEntry('softkey1 label','Test');
$configuration->addEntry('softkey1 type','xml');
$configuration->setTriggerDestroyOnExit();
$configuration->setBeep();
$configuration->output();.
```

In this example, the label and the type of the softkey 1 are changed.

12.2 AastraIPPhoneDirectory()

This class allows you to create a XML Directory object on a 55i/57i/57iCT.

Include

AastraIPPhoneDirectory.class.php

Methods

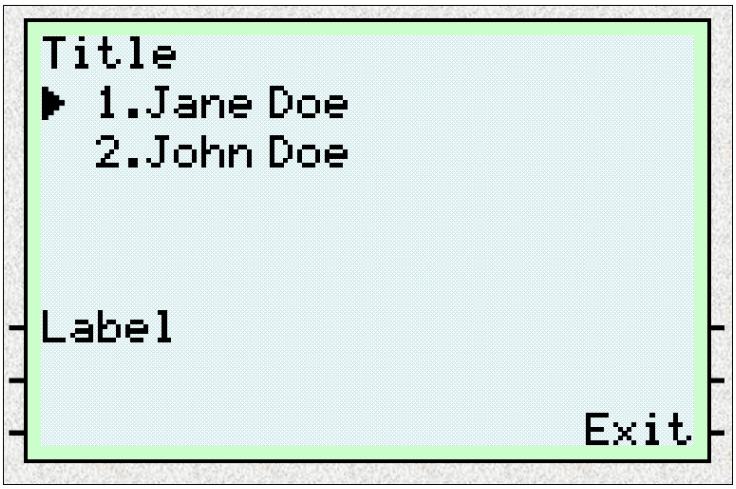
- setTitle(Title) to setup the title of an object
- setTitleWrap() to set the title to be wrapped on 2 lines (optional)
- setDestroyOnExit() to set destroyOnExit parameter to 'yes', 'no' by default (optional)
- setCancelAction(uri) to set the cancel parameter with the URI to be called on Cancel (optional)
- setBeep() to enable a notification beep with the object (optional)

- setLockIn() to set the Lock-in tag to 'yes' (optional)
- setTimeout(timeout) to define a specific timeout for the XML object (optional)
- addSoftkey(index,label,uri,icon_index) to add custom softkeys to the object (optional)
- addIcon(index,icon) to add custom icons to the object (optional)
- setRefresh(timeout,URL) to add Refresh parameters to the object (optional)
- output() to display the object
- setNext(next) to set URI of the next page (optional)
- setPrevious(previous) to set URI of the previous page (optional)
- addEntry(name,phone) to add an element in the list to be displayed, at least one is needed.
- natsortbyname() to order the list

Example

```
require_once('AastraIPPhoneDirectory.class.php');
$directory = new AastraIPPhoneDirectory();
$directory->setTitle('Title');
$directory->setNext('http://myserver.com/script.php?page=2');
$directory->setPrevious('http://myserver.com/script.php?page=0');
$directory->setDestroyOnExit();
$directory->addEntry('John Doe', '200');
$directory->addEntry('Jane Doe', '201');
$directory->natsortByName();
$directory->addSoftkey('1', 'Label',
'http://myserver.com/script.php?action=1');
$directory->addSoftkey('6', 'Exit', 'SoftKey:Exit');
$directory->output();
```

Output



12.3 AastraIPPhoneExecute()

This class allows you to create a XML PhoneExecute object.

Include

AastraIPPhoneExecute.class.php'

Methods

- output() to display the object
- setBeep() to enable a notification beep with the object (optional)
- setTriggerDestroyOnExit() to set the triggerDestroyOnExit tag to "yes" (optional)
- addEntry(url) to add an action to be executed..

Example

```
require_once('AastraIPPhoneExecute.class.php');
$execute = new AastraIPPhoneExecute();
$execute->addEntry('http://myserver.com/script.php?choice=2');
$execute->addEntry('Command: Reset');
$execute->output();
```

12.4 AastraIPPhoneFormattedTextScreen()

This class allows you to create a XML FormattedTextScreen object.

Include

AastraIPPhoneFormattedTextScreen.class.php

Methods

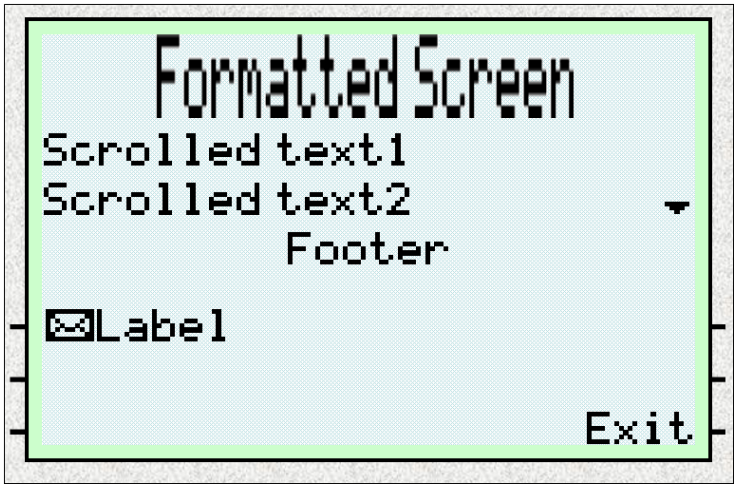
- setDestroyOnExit() to set destroyOnExit parameter to 'yes', 'no' by default (optional)
- setCancelAction(uri) to set the cancel parameter with the URI to be called on Cancel (optional)
- setDoneAction(uri) to set the URI to be called when the user selects the default "Done" key (optional)
- setBeep() to enable a notification beep with the object (optional)
- setLockIn() to set the Lock-in tag to 'yes' (optional)
- setAllowAnswer() to set the allowAnswer tag to 'yes' (optional)
- setTimeout(timeout) to define a specific timeout for the XML object (optional)
- addSoftkey(index, label, uri) to add custom softkeys to the object (optional)
- addIcon(index,icon) to add custom icons to the object (optional)
- setRefresh(timeout,URL) to add Refresh parameters to the object (optional)
- output() to display the object
- generate() to return the object content as a string for debug purpose
- addLine(text,size,align) to add a formatted line
- setScrollStart(height) to define the beginning of the scrolling section and its height
- setScrollEnd() to define the end of the scrolling section

Example

```
require_once('AastraIPPhoneFormattedTextScreen.class.php');
$ftext = new AastraIPPhoneFormattedTextScreen();
```

```
$ftext->setDestroyOnExit();
$ftext->addLine('Formatted Screen','double','center');
$ftext->setScrollStart('2');
$ftext->addLine('Scrolled text1');
$ftext->addLine('Scrolled text2');
$ftext->addLine('Scrolled text3');
$ftext->addLine('Scrolled text4');
$ftext->addLine('Scrolled text5');
$ftext->setScrollEnd();
$ftext->addLine('Footer',NULL,'center');
$ftext->addSoftkey('1', 'Label',
'http://myserver.com/script.php?action=1','1');
$ftext->addSoftkey('6', 'Exit', 'SoftKey:Exit');
$ftext->addIcon('1', 'Icon:Envelope');
$ftext->output();
$menu->output();
```

Output



12.5 AastraIPPhoneImageMenu()

This class allows you to create a XML ImageMenu object on 55i/57i/57iCT.

Include

AastraIPPhoneImageMenu.class.php

Methods

- setDestroyOnExit() to set destroyOnExit parameter to "yes" (optional)
- setCancelAction(uri) to set the cancel parameter with the URI to be called on Cancel (optional)
- setBeep() to enable a notification beep with the object (optional)
- setTimeout(timeout) to define a specific timeout for the XML object (optional)
- setLockIn() to set the Lock-in tag to 'yes' (optional)
- addSoftkey(index,label,uri,icon_index) to add custom softkeys to the object (optional)
- setRefresh(timeout,URL) to add Refresh parameters to the object (optional)
- output() to display the object

- generate() to return the object content as a string for debug purpose
- setImage(image)to define the image to be displayed
- setAlignment(vertical,horizontal) to define image alignment
- setSize(height,width) to define image size
- setURIBase(uriBase) to define the base URI for the selections
- addURI(key,uri) to add a selection key with its URI

Example

```
require_once('AastraIPPhoneImageMenu.class.php');
$imagem = new AastraIPPhoneImageMenu();
$imagem->setDestroyOnExit();
$imagem->setSize(40,144);
$imagem-
>setImage('fffffffc02ffffffee4ffffbfffc05fffe7ff7a7fffffffeffeebd
7fffffea6bcffffe796f3feff6fa289f0a86f4866fa20df42414595dd0134f8037
ed1637f0e2522b2dd003b6eb936f05fffbd4f4107bba6eb0080e93715000010b754
001281271408c640252081b1b22500013c5c66201368004e04467520dc11067152b
82094d418e10024720580549478010500260153002013140002 0a05c91088b002b0
8c21c0000c200000001fe8000000000000000001c0000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000002004100000400830000 0ff085000000
00c900000000710000000000000001401400000140140000014014000001401400 0
0014014000000000000007c0ff00000c3088000008108800000810880000 0c3070
0000062000000000003f000001e020000003302000000210000000003301e000001e
033000000002100000 3f033000002001e000002000000000001e000c03fc33003c
013021007c02101201f00330ff03f001e000039000003e039001e00103f00330010
1f8021003007c03303f003c01e000000c00001e001c03f033007802002100f00200
2103e000001203c401702003cc0290ff039c02902101fc02b000007c03f01a003c0
20039018c0ff02d03c402102703c400001203ec01e000026402b000026402900002
6c029000027c01a000033800000003380000000310000000300000000030003f00
003fc03000003fc02000003fc020000030001f000030000000030001e000030002
b000030002900003fc02900003fc01a00003f000000031003000003 1c01e000031
f003000033f81e00003f383000001e081e000008c003000003c01e00000fc030000
01f000000003d001a000039003900003900 2d00003f002700000f8012000007c000
000001c000000004000000000000000000000000000');
$imagem->addURI('1','http://myserver.com?choice=1');
$imagem->addURI('2','http://myserver.com?Choice=2');
$imagem->addSoftkey('1', 'Label',
'http://myserver.com/script.php?action=1','1');
$imagem->addSoftkey('6', 'Exit', 'SoftKey:Exit');
$imagem->addIcon('1', 'Icon:Envelope');
$imagem->addIcon('2', 'FFFF0000FFFF0000');
$imagem->output();
```

Output

12.6 AastraIPPhoneImageScreen()

This class allows you to create a XML ImageScreen object.

Include

AastraIPPhoneImageScreen.class.php

Methods

- setDestroyOnExit() to set destroyOnExit parameter to 'yes', 'no' by default (optional)
- setCancelAction(uri) to set the cancel parameter with the URI to be called on Cancel (optional)
- setBeep() to enable a notification beep with the object (optional)
- setLockIn() to set the Lock-in tag to 'yes' (optional)
- setTimeout(timeout) to define a specific timeout for the XML object (optional)
- addSoftkey(index,label,uri,icon_index) to add custom softkeys to the object (optional)
- addIcon(index,icon) to add custom icons to the object (optional)
- setRefresh(timeout,URL) to add Refresh parameters to the object (optional)
- output() to display the object
- generate() to return the object content as a string for debug purpose
- setImage(image)to define the image to be displayed
- setAlignment(vertical,horizontal) to define image alignment
- setSize(height,width) to define image size

Example

```
require_once('AastraIPPhoneImageScreen.class.php');
$images = new AastraIPPhoneImageScreen();
$images->setDestroyOnExit();
$images->setSize(40,40);
$images-
>setImage('ffffffc02ffffffee4ffffbfffc05fffe7ff7a7ffffffffeffeebd
7fffffea6bcffffe796f3feff6fa289f0a86f4866fa20df42414595dd0134f8037
```

```
ed1637f0e2522b2dd003b6eb936f05fffbd4f4107bba6eb0080e93715000010b754
001281271408c640252081b1b22500013c5c66201368004e04467520dc11067152b
82094d418e10024720580549478010500260153002093140002 0ac5c91088b0f2b0
8c21c07d0c2006009fdfe81f80efe0107fe0fb1c3ffff8ffc3fffef8f7febffbfcf
87ffbff64');
$images->addSoftkey('1', 'Mail',
'http://myserver.com/script.php?action=1','1');
$images->addSoftkey('6', 'Exit', 'SoftKey:Exit');
$images->addIcon('1', 'Icon:Envelope');
$images->output();
```

Output

12.7 AastraIPPhoneInputScreen() – Single Input field

This class allows you to create a XML InputScreen object.

Include

AastraIPPhoneInputScreen.class.php'

Methods

- setTitle(Title) to setup the title of an object (optional)
- setTitleWrap() to set the title to be wrapped on 2 lines (optional)
- setDestroyOnExit() to set DestroyonExit parameter to 'yes', 'no' by default (optional)
- setCancelAction(uri) to set the cancel parameter with the URI to be called on Cancel (optional)
- setBeep() to enable a notification beep with the object (optional)
- setLockIn() to set the Lock-in tag to 'yes' (optional)
- setAllowAnswer() to set the allowAnswer tag to 'yes' (optional)
- setTimeout(timeout) to define a specific timeout for the XML object (optional)
- addSoftkey(index,label,uri,icon_index) to add custom softkeys to the object (optional)
- addIcon(index,icon) to add custom icons to the object (optional)
- generate() to return the object content as a string

- output() to display the object
- setURL(url) to set the URL to called after the input
- setType(type) to set type of input ('IP', 'string', 'number', 'dateUS', 'dateInt', 'timeUS', 'timeInt'), 'string' by default
- setDefault(default) to set default value for the input (optional)
- setParameter(param) to set the parameter name to be parsed after the input
- setPassword() to set the Password parameter to 'yes', 'no' by default (optional)
- setNotEditable() to set the editable parameter to 'no', 'yes' by default (optional)
- setEditable() is now replaced by setNotEditable but kept for compatibility reasons (optional)
- setPrompt(prompt) to set the prompt to be displayed for the input.

Example

```
require_once('AastraIPPhoneInputScreen.class.php');
$input = new AastraIPPhoneInputScreen();
$input->setTitle('Title');
$input->setPrompt('Enter your password');
$input->setParameter('param');
$input->setType('string');
$input->setURL('http://myserver.com/script.php');
$input->setPassword();
$input->setDestroyOnExit();
$input->setDefault('Default');
$input->output();
```

Output

12.8 AastraIPPhoneInputScreen() – Multiple Input fields

This class allows you to create a XML InputScreen object with multiple input fields (55i/57i and 57iCT)

Include

AastraIPPhoneInputScreen.class.php'

Methods

- setTitle(Title) to setup the title of an object (optional)
- setTitleWrap() to set the title to be wrapped on 2 lines (optional)
- setDestroyOnExit() to set DestroyonExit parameter to 'yes', 'no' by default (optional)
- setCancelAction(uri) to set the cancel parameter with the URI to be called on Cancel (optional)
- setBeep() to enable a notification beep with the object (optional)
- setLockIn() to set the Lock-in tag to 'yes' (optional)
- setTimeout(timeout) to define a specific timeout for the XML object (optional)
- addSoftkey(index,label,uri,icon_index) to add custom softkeys to the object (optional)
- addIcon(index,icon) to add custom icons to the object (optional)
- generate() to return the object content as a string
- output() to display the object
- setURL(url) to set the URL to called after the input
- setType(type) to set the default type of input ('IP', 'string', 'number', 'dateUS', 'dateInt', 'timeUS', 'timeInt'), 'string' by default
- setDefault(default) to set default default value for the input (optional)
- setParameter(param) to set the default parameter name to be parsed after the input
- setPassword() to set the default Password parameter to 'yes', 'no' by default (optional)
- setNotEditable() to set the default editable parameter to 'no', 'yes' by default (optional)
- setEditable() is now replaced by setNotEditable but kept for compatibility reasons (optional)
- setPrompt(prompt) to set the default prompt to be displayed for the input.
- setDefaultIndex(index) to define the field index the object will use to start (optional) default is 1
- setDisplayMode(display) to define the aspect of the display, normal/condensed (optional) default is normal.
- addField(type) to add an input field and setting its type ('IP', 'string', 'number', 'dateUS', 'dateInt', 'timeUS', 'timeInt' or 'empty'), if the type is an empty string, the type is inherited from the main object.
- setFieldPassword(password) to set the password mode for the input field ('yes', no'), overrides the value set by setPassword for the field
- setFieldEditable(editable) to set the input field editable mode ('yes', no'), overrides the value set by setEditable or setNotEditable for the field
- setFieldParameter(parameter) to set the parameter name to be parsed after the global input, overrides the value set by setParameter for the field
- setFieldPrompt(prompt)to set the prompt to be displayed for the input field, overrides the value set by setPrompt for the field
- setFieldDefault(default) to set default value for the input field, overrides the value set by setDefault for the field

- setFieldSelection(selection) to add a selection tag to the URI passed backed to the XML application when the user validate the object while editing the field.
- addFieldSoftkey(index,label,uri,icon) to add custom softkeys to the input field, overrides the softkeys set by addSoftkey.

Example

```
require_once('AastraIPPhoneInputScreen.class.php');
$input = new AastraIPPhoneInputScreen();
$input->setTitle('Restricted application');
$input->setDisplayMode('condensed');
$input->setURL($XML_SERVER);
$input->setDestroyOnExit();
$input->addSoftkey('5', 'Done', 'SoftKey:Submit');
$input->addSoftkey('6', 'Exit', 'SoftKey:Exit');
$input->addField(‘empty’);
$input->addField(‘string’);
$input->setFieldSelection('1');
$input->setFieldPrompt('Username:');
$input->setFieldParameter('user');
$input->setFieldSelection('1');
$input->addFieldSoftkey('3', 'ABC', 'SoftKey:ChangeMode');
$input->addField(‘number’);
$input->setFieldPassword('yes');
$input->setFieldPrompt('Password:');
$input->setFieldParameter('password');
$input->setFieldSelection('2');
$input->output();
```

Output

12.9 AastraIPPhoneStatus()

This class allows you to create a XML PhoneStatus object.

Include

AastraIPPhoneStatus.class.php

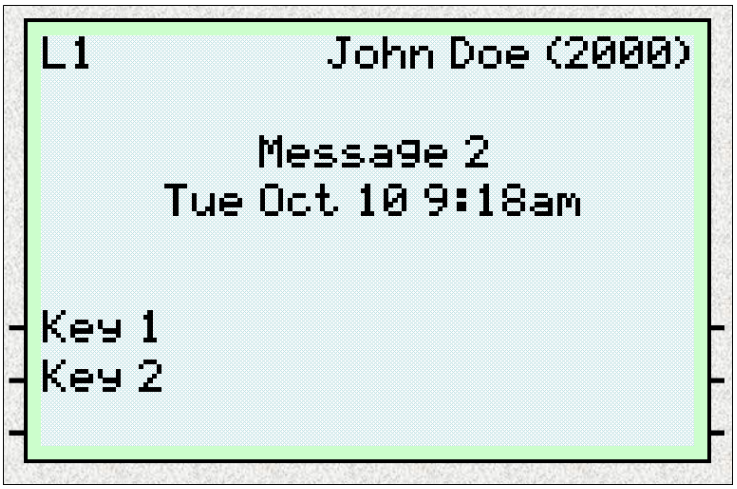
Methods

- output() to display the object
- generate() to return the object content as a string for debug purpose
- setBeep() to enable a notification beep with the object (optional)
- setSession(session) to setup the session ID
- setTriggerDestroyOnExit() to set the triggerDestroyOnExit tag to "yes" (optional)
- addEntry(index,message,type,timeout) to add a message to be displayed on the idle screen..

Example

```
require_once('AastraIPPhoneStatus.class.php');
$status = new AastraIPPhoneStatus();
$status->setSession('Session');
$status->setBeep();
$status->addEntry('1','Message 1');
$status->addEntry('2','Alert',’alert’,5);
$status->output();
```

Output



12.10 AastraIPPhoneTextMenu()

This class allows you to create a XML TextMenu object.

Include

AastraIPPhoneTextMenu.class.php

Methods

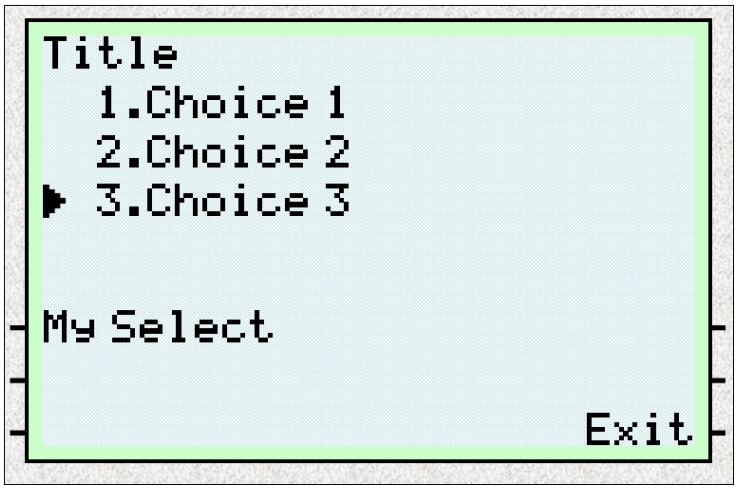
- setTitle(Title) to setup the title of an object
- setTitleWrap() to set the title to be wrapped on 2 lines (optional)
- setCancelAction(uri) to set the cancel parameter with the URI to be called on Cancel (optional)
- setDestroyOnExit() to set DestroyOnExit parameter to "yes" (optional)
- setBeep() to enable a notification beep with the object (optional)

- setLockIn() to set the Lock-in tag to 'yes' (optional)
- setAllowAnswer() to set the allowAnswer tag to 'yes' (optional)
- setTimeout(timeout) to define a specific timeout for the XML object (optional)
- addSoftkey(index,label,uri,icon_index) to add custom softkeys to the object (optional)
- addIcon(index,icon) to add custom icons to the object (optional)
- setRefresh(timeout,URL) to add Refresh parameters to the object (optional)
- output() to display the object
- generate() to return the object content as a string for debug purpose
- setDefaultIndex(index) to set the default selection in the list (optional)
- setStyle(style) to set the style of the list numbered/none/radio (optional)
- addEntry(name,url,selection,icon) to add an element in the list to be displayed, at least one is needed.
- natsortbyname() to order the list

Example 1

```
require_once('AastraIPPhoneTextMenu.class.php');
$menu = new AastraIPPhoneTextMenu();
$menu->setTitle('Title');
$menu->setDestroyOnExit();
$menu->setDeFaultIndex('3');
$menu->addEntry('Choice 2',
'http://myserver.com/script.php?choice=2', 'Value=2');
$menu->addEntry('Choice 1',
'http://myserver.com/script.php?choice=1', 'Value=1');
$menu->addEntry('Choice 3',
'http://myserver.com/script.php?choice=3', 'Value=3');
$menu->natsortByName();
$menu->addSoftkey('1', 'My Select',
'http://myserver.com/script.php?action=1');
$menu->addSoftkey('6', 'Exit', 'SoftKey:Exit');
$menu->output();
```

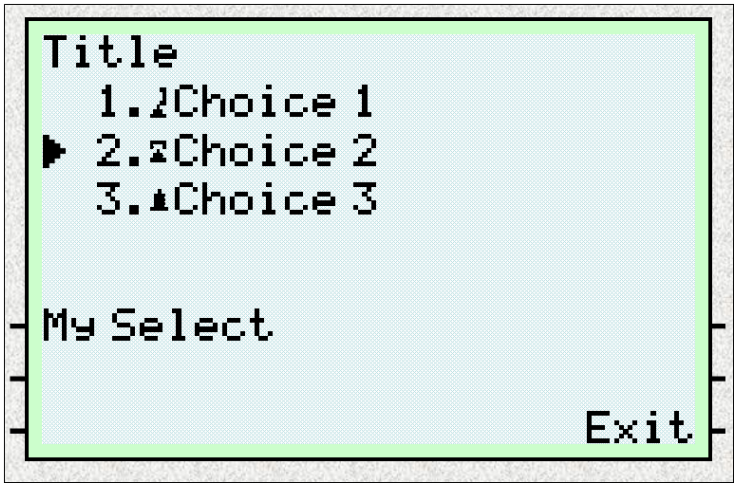
Output



Example 2

```
require_once('AastraIPPhoneTextMenu.class.php');
$menu = new AastraIPPhoneTextMenu();
$menu->setTitle('Title');
$menu->setDestroyOnExit();
$menu->setDeFaultIndex('2');
$menu->addEntry('Choice 2',
'http://myserver.com/script.php?choice=2', 'Value=2','1');
$menu->addEntry('Choice 1',
'http://myserver.com/script.php?choice=1', 'Value=1','2');
$menu->addEntry('Choice 3',
'http://myserver.com/script.php?choice=3', 'Value=3','3');
$menu->natsortByName();
$menu->addSoftkey('1', 'My Select',
'http://myserver.com/script.php?action=1');
$menu->addSoftkey('6', 'Exit', 'SoftKey:Exit');
$menu->addIcon('1', 'Icon:PhoneOnHook');
$menu->addIcon('2', 'Icon:PhoneOffHook');
$menu->addIcon('3', 'Icon:PhoneRinging');
$menu->output();
```

Output



12.11 AastraIPPhoneTextScreen()

This class allows you to create a XML TextScreen object.

Include

AastraIPPhoneTextScreen.class.php

Methods

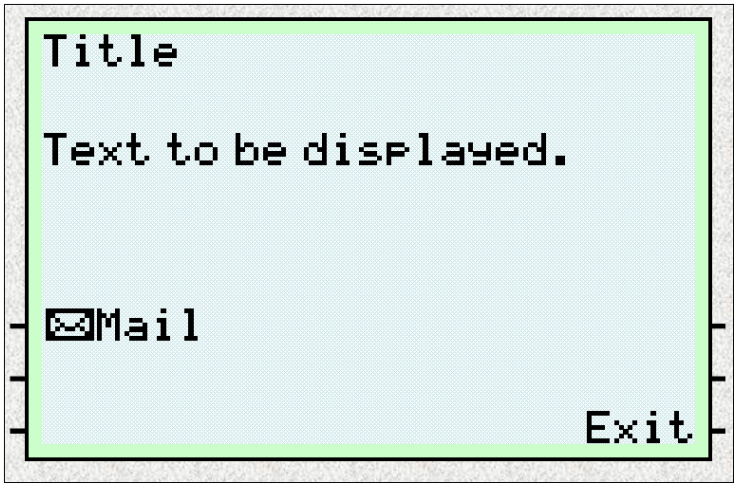
- setTitle(Title) to setup the title of an object
- setTitleWrap() to set the title to be wrapped on 2 lines (optional)
- setCancelAction(uri) to set the cancel parameter with the URI to be called on Cancel (optional)
- setDoneAction(uri) to set the URI to be called when the user selects the default "Done" key (optional)
- setDestroyOnExit() to set destroyOnExit parameter to 'yes', 'no' by default (optional)

- setBeep() to enable a notification beep with the object (optional)
- setLockIn() to set the Lock-in tag to 'yes' (optional)
- setAllowAnswer() to set the allowAnswer tag to 'yes' (optional)
- setTimeout(timeout) to define a specific timeout for the XML object (optional)
- addSoftkey(index,label,uri,iconindex) to add custom softkeys to the object (optional)
- addIcon(index,icon) to add custom icons to the object (optional)
- setRefresh(timeout,URL) to add Refresh parameters to the object (optional)
- output() to display the object
- generate() to return the object content as a string for debug purpose
- setText(text) to set the text to be displayed..

Example

```
require_once('AastraIPPhoneTextScreen.class.php');
$text = new AastraIPPhoneTextScreen();
$text->setTitle('Title');
$text->setText('Text to be displayed.');
$text->setDestroyOnExit();
$text->addSoftkey('1', 'Mail',
'http://myserver.com/script.php?action=1','1');
$text->addSoftkey('6', 'Exit', 'SoftKey:Exit');
$text->addIcon('1', 'Icon:Envelope');
$text->output();
```

Output



13 Appendix C: Dynamic Parameters

The following table lists the configuration parameters that can be modified by a PhoneConfiguration object without a reboot.

Parameter	Comments
softkeyN type	Changes to subscriptions (eg BLF or BLA) require a reboot.
softkeyN label	
softkeyN value	
softkeyN line	
softkeyN states	
prgkeyN type	Changes to subscriptions (eg BLF or BLA) require a reboot.
prgkeyN name	
prgkeyN value	
prgkeyN line	
live dialpad	
tone set	
ring tone	
audio mode	
language	
ringback timeout	
headset tx gain	
headset sidetone gain	
handset tx gain	
handset sidetone gain	
handsfree tx gain	
line1 ring tone	
line2 ring tone	
line3 ring tone	
line4 ring tone	
line5 ring tone	
line6 ring tone	
line7 ring tone	
line8 ring tone	
line9 ring tone	
suppress dtmf playback	

Parameter	Comments
redial disabled	This parameter is dynamic so a user can't access it or add to it. However, you need to reboot the phone to clear the list.
call transfer disabled	
conference disabled	
directory disabled	This parameter is dynamic so a user can't access it or add to it. However, you need to reboot the phone to clear the list.
callers list disabled	This parameter is dynamic so a user can't access it or add to it. However, you need to reboot the phone to clear the list.
options password enabled	
time server disabled	
time reserved	
dst config	
time server1	
time server2	
time server3	
time format	
date format	
time zone name	
time zone code	
time zone minutes	
dst minutes	
dst start relative date	
dst start month	
dst start day	
dst start week	
dst start hour	
dst end relative date	
dst end month	
dst end day	
dst end week	
dst end hour	

Parameter	Comments
tftp server	
alternate tftp server	
use alternate tftp server	
admin password	
user password	
sip nat ip	
sip nat port	
sip dial plan	
sip dial plan terminator	
sip digit timeout	
sip blf subscription period	
sip registration retry timer	
sip forward number	
sip forward mode	
sip ring number	
sip vmail	
sip dtmf method	
sip lineN forward number	
sip lineN forward mode	
sip lineN ring number	
sip lineN vmail	
sip lineN dtmf method	
sip intercom type	
sip intercom prefix code	
sip intercom mute mic	
sip intercom line	
sip allow auto answer	
sip silence suppression	
sip send mac	
sip send line	
xml application URI	
xml application title	
xml beep notification	
action uri registered	

Parameter	Comments
action uri incoming	
action uri outgoing	
action uri onhook	
action uri offhook	
action uri startup	
directed call pickup	
play a ring splash	
map redial key to	
map conf key to	
download protocol	
ftp server	
ftp username	
ftp password	
http server	
http path	
directory 1	You need to reboot the phone to download new directories.
directory 2	
directory script	
auto resync time	
auto resync mode	

14 Appendix D: XML Self-Configuration

Aastra provides here a possible implementation for the self-configuration, this is just an example source code not supported by Aastra Telecom.

In this implementation targeted for Asterisk (trixbox implementation), the chosen policy is to have all the extensions pre-configured in the Asterisk database and use

- the extension
- the voice mail password

for the credentials to authenticate the user and link the MAC address of the phone to an extension.

The script checks the credentials but also uses a log file to trace the extensions that are already used (startup.cfg)

Note:

- TFTP server root directory is located at /tftpboot
- XML script and all the related files are located at the root directory of the HTTP server (typically /var/www/html) under the directory “startup“.

aastra.cfg (typically located at /tftpboot)

Note: In this example, the Asterisk server is located at 192.168.0.110 and is also the TFTP server.

```
# Setup DHCP mode
dhcp: 1

# Setup TFTP server address
tftp server: 192.168.0.110

# Time server
time server disabled: 0
time server1: pool.ntp.org

# Startup URI
action uri startup: http://192.168.0.110/startup/startup.php
```

gen-aastra script shell to generate the aastra.cfg file

```
#!/bin/sh
# To put on /usr/local/sbin
echo ""
echo "------------------------------------------------------------
-"
echo "Creating a default config file for Aastra phones"
echo "------------------------------------------------------------
-"
echo ""
echo "Creating /tftpboot/aastra.cfg..."
echo ""

IFCONFIG=`which ifconfig 2>/dev/null||echo /sbin/ifconfig`
```

```
IPADDR=`$IFCONFIG eth0|gawk '/inet addr/{print $2}'|gawk -F:
'{print $2}'`

cat > /tftpboot/aastra.cfg <<EOF
# Setup DHCP mode
dhcp: 1

# Setup TFTP server address
tftp server: $IPADDR

# Time server
time server disabled: 0
time server1: pool.ntp.org

# Startup URI
action uri startup: http://$IPADDR/xml/startup/startup.php

EOF

chmod 666 /tftpboot/aastra.cfg

echo "Created /tftpboot/aastra.cfg using $IPADDR for the proxy. If
the"
echo "IP address of your Asterisk system changes run this script
again and reboot."
echo "Reboot your Aastra phones by disconnecting the power to the
phone."
echo ""
```

Template files for the phones (typically located at /var/www/html/startup)

The following file is a template used to create the MAC.cfg file, there must be one file for each Aastra phone.

- Aastra53i.cfg for the 53i
- Aastra55i.cfg for the 55i
- Aastra57i.cfg for the 57i
- Aastra57iCT.cfg for the 57iCT

In these template files, you can use variables that will be replaced by the corresponding value by the script.

- $$AA_SIPAUTHNAME_AA$$ for the user SIP authentication name
- $$AA_SIPSECRET_AA$$ for the user SIP secret
- $$AA_SIPUSERNAME_AA$$ for the user SIP username
- $$AA_SIPCALLERID_AA$$ for the user SIP caller-id
- $$AA_PROXY_SERVER_AA$$ for the name/IP address of the proxy server
- $$AA_REGISTRAR_SERVER_AA$$ for the name/IP address of the registrar server

The template files can of course be completed with extra parameters (timezone, softkeys, …).

```
# SIP Lines
sip line1 auth name: $$AA_SIPAUTHNAME_AA$$
sip line1 password: $$AA_SIPSECRET_AA$$
sip line1 user name: $$AA_SIPUSERNAME_AA$$
sip line1 display name: $$AA_SIPCALLERID_AA$$
sip line1 screen name: $$AA_SIPCALLERID_AA$$
```

```
sip line1 proxy ip: $$AA_PROXY_SERVER_AA$$
sip line1 proxy port: 5060
sip line1 registrar ip: $$AA_REGISTRAR_SERVER_AA$$
sip line1 registrar port: 5060
sip line1 vmail: *98
sip line1 mode: 0

# Action URI
action uri startup:
```

startup.php (located at /var/www/html/startup)

```
<?php

#
# Startup.php
# Sample script for self configuration with Digium/Asterisk platform.
# Copyright Aastra Telecom 2007
#

#
# Includes
#
require_once('include/config.inc.php');
include_once('include/backend.inc.php');

#
# Aastra_decode_HTTP_header
#
# Returns an array
#   0 Phone Type
#   1 Phone MAC Address
#   2 Phone firmware version
#   3 IP address
#

function Aastra_decode_HTTP_header()
{
$user_agent=$_SERVER["HTTP_USER_AGENT"];
if(stristr($user_agent,"Aastra"))
        {
        $value=preg_split("/ MAC:/",$user_agent);
        $fin=preg_split("/ /",$value[1]);
        $value[1]=preg_replace("/\-/","",$fin[0]);
        $value[2]=preg_replace("/V:/","",$fin[1]);
        }
else
        {
        $value[0]="MSIE";
        $value[1]="NA";
        $value[2]="NA";
        }

$value[3]=$_SERVER["REMOTE_ADDR"];

return($value);
}

#
# lookup_config_file(extension)
#
# Returns 1 if extension already in use
#

function lookup_config_file($extension)
{
```

```
$config="startup.cfg";

# Init return
$return=0;

# Read config file
$array = @parse_ini_file($config, true);

# Test MAC address
if($array[$extension]['mac']!="") $return=1;

return($return);
}

#
# update_config_file(extension,mac,ip,model)
#
# Update the config file with the new extension
#

function update_config_file($extension,$mac,$ip,$model)
{
$config="startup.cfg";

# Read config file
$array = @parse_ini_file($config, true);
if($array==NULL) $array=array();

# Update value
$array[$extension]['mac']=$mac;
$array[$extension]['ip']=$ip;
$array[$extension]['model']=$model;

# Update config file
reset($array);
$handle = @fopen($config, "w");
if($handle)
        {
        while ($v = current($array))
                {
                fputs($handle,"[".key($array)."]"."\n");
                fputs($handle,"mac=".$v['mac']."\n");
                fputs($handle,"ip=".$v['ip']."\n");
                fputs($handle,"model=".$v['model']."\n\n");
                next($array);
                }
        fclose($handle);
        }
}

#
# create_mac(extension,mac,username,secret,callerid,model,version)
#
# Creates the MAC.cfg file for the user.
#

function create_mac($mac,$extension,$username,$secret,$callerid,$model,$version)
{
Global $AA_PROXY_SERVER,$AA_REGISTRAR_SERVER;

$value=preg_split("/ /",$callerid);
$result=$value[0]." ".$value[1];
$result=preg_replace("/</","(",$result);
$result=preg_replace("/>/",")",$result);

# Prepare replace strings
$search=array('/\$\$AA_SIPAUTHNAME_AA\$\$/','/\$\$AA_SIPSECRET_AA\$\$/','/\$\$AA_SI
PUSERNAME_AA\$\$/','/\$\$AA_SIPCALLERID_AA\$\$/','/\$\$AA_PROXY_SERVER_AA\$\$/','/\
$\$AA_REGISTRAR_SERVER_AA\$\$/');
$replace=array($username,$secret,$extension,$result,$AA_PROXY_SERVER,$AA_REGISTRAR_
SERVER);
```

```
$read = @fopen($model.".cfg", "r");
if($read)
	{
	$write = @fopen("/tftpboot/".$mac.".cfg", "w");
	if($write)
		{
		# Create file header
		while($line=fgets($read,200))
			{
			$line = preg_replace($search, $replace, $line);
			fputs($write,$line);
			}
		fputs($write,"\n");
		fclose($write);
		}
	fclose($read);
	}
}

##########################################
# GLOBAL VARIABLES

$XML_SERVER = "http://".$_SERVER['SERVER_ADDR'].$_SERVER['SCRIPT_NAME'];
$AA_PROXY_SERVER = $_SERVER['SERVER_ADDR'];
$AA_REGISTRAR_SERVER = $_SERVER['SERVER_ADDR'];
$location = "/etc/asterisk/";

##########################################
# Retrieve parameters
$extension=$_GET["extension"];
$password=$_GET["password"];
$action=$_GET["action"];
$step=$_GET["step"];

# Set content type
header("Content-Type: text/xml");

if($action=="countdown")
	{
	# Display Reboot message
	$output = "<AastraIPPhoneTextScreen destroyOnExit=\"yes\">\n";
	$output .= "<Title>Reboot needed</Title>\n";
	$output .= "<Text>Your phone will reboot in ".$step." seconds.</Text>\n";
	$step--;
	$output .= "</AastraIPPhoneTextScreen>\n";
	header("Content-Type: text/xml");
	header("Content-Length: ".strlen($output));
	if($step==0) header("Refresh: 1; url=".$XML_SERVER."?action=reboot");
	else header("Refresh: 1; url=".$XML_SERVER."?action=countdown&step=".$step);
	echo $output;
	exit;
	}

if($action=="reboot")
	{
	$output = "<AastraIPPhoneExecute>\n";
	$output .= "<ExecuteItem URI=\"Command: Reset\"/>\n";
	$output .= "</AastraIPPhoneExecute>\n";
	header("Content-Length: ".strlen($output));
	echo $output;
	exit;
	}

# Input Extension
if ($extension=="")
	{
	$output = "<AastraIPPhoneInputScreen type=\"number\" LockIn=\"yes\">\n";
	$output .= "<Title>Initial startup</Title>\n";
	$output .= "<Prompt>Enter Extension</Prompt>\n";
	$output .= "<URL>$XML_SERVER</URL>\n";
```

```
        $output .= "<Parameter>extension</Parameter>\n";
        $output .= "<Default></Default>\n";
        $output .= "</AastraIPPhoneInputScreen>\n";
        header("Content-Length: ".strlen($output));
        echo $output;
        exit;
        }

# Input Password
if ($password=="")
        {
        $output = "<AastraIPPhoneInputScreen type=\"number\" password=\"yes\"
LockIn=\"yes\" destroyOnExit=\"yes\">\n";
        $output .= "<Title>Initial startup</Title>\n";
        $output .= "<Prompt>Enter Password</Prompt>\n";
        $output .= "<URL>$XML_SERVER?extension=$extension</URL>\n";
        $output .= "<Parameter>password</Parameter>\n";
        $output .= "<Default></Default>\n";
        $output .= "</AastraIPPhoneInputScreen>\n";
        header("Content-Length: ".strlen($output));
        echo $output;
        exit;
        }

# IF authentication failed
if (!$userinfo = verify_user($extension, $password))
        {
        # Display error
        $output = "<AastraIPPhoneTextScreen LockIn=\"yes\"
destroyOnExit=\"yes\">\n";
        $output .= "<Title>Authentication failed</Title>\n";
        $output .= "<Text>Wrong user and password.</Text>\n";
        $output .= "</AastraIPPhoneTextScreen>\n";
        header("Content-Type: text/xml");
        header("Content-Length: ".strlen($output));
        echo $output;
        exit;
        }

# IF already configured
if(lookup_config_file($extension)==1)
        {
        # Display error
        $output = "<AastraIPPhoneTextScreen LockIn=\"yes\"
destroyOnExit=\"yes\">\n";
        $output .= "<Title>Error</Title>\n";
        $output .= "<Text>Extension already in use.</Text>\n";
        $output .= "</AastraIPPhoneTextScreen>\n";
        header("Content-Type: text/xml");
        header("Content-Length: ".strlen($output));
        echo $output;
        exit;
        }

# Get all the user data
$sip_array = parse_ini_file($location."sip_additional.conf", true);

# If user not found
if ($sip_array[$extension]==NULL)
        {
        # Display error
        $output = "<AastraIPPhoneTextScreen LockIn=\"yes\"
destroyOnExit=\"yes\">\n";
        $output .= "<Title>Internal error</Title>\n";
        $output .= "<Text>Extension is not provisioned.</Text>\n";
        $output .= "</AastraIPPhoneTextScreen>\n";
        header("Content-Type: text/xml");
        header("Content-Length: ".strlen($output));
        echo $output;
        exit;
        }
```

```
else
        {
        # Collect data
        $username=$sip_array[$extension]['username'];
        $secret=$sip_array[$extension]['secret'];
        $callerid=$sip_array[$extension]['callerid'];
        }

# Get MAC address and type of phone
$value=Aastra_decode_HTTP_header();

# Create mac.cfg
create_mac($value[1],$extension,$username,$secret,$callerid,$value[0],$value[2]);

# Update config file
update_config_file($extension,$value[1],$value[3],$value[0]);

# Create Reboot screen
$output = "<AastraIPPhoneTextScreen destroyOnExit=\"yes\">\n";
switch($value[0])
        {
        case "Aastra53i":
                $output = "<AastraIPPhoneTextScreen>\n";
                $output .= "<Title>REBOOT</Title>\n";
                $output .= "<Text>Reboot</Text>\n";
                $output .= "</AastraIPPhoneTextScreen>\n";
                header("Refresh: 1; url=".$XML_SERVER."?action=reboot");
                break;

        default:
                $output = "<AastraIPPhoneTextScreen>\n";
                $output .= "<Title>Reboot needed</Title>\n";
                $output .= "<Text>Your phone will reboot in 5 seconds.</Text>\n";
                $output .= "</AastraIPPhoneTextScreen>\n";
                header("Refresh: 1; url=".$XML_SERVER."?action=countdown&step=4");
                break;
        }

# Display XML Object
header("Content-Type: text/xml");
header("Content-Length: ".strlen($output));
echo $output;
exit;
?>
```

15 Appendix E: Sample Applications for Asterisk / Digium Open PBX

15.1 Introduction

These examples are available on the Aastra Web site http://www.aastratelecom.com.

Description of the applications:

- Directory, the script parses the SIP users and the IAX users configured on your Asterisk server and display them as a list where you can dial from. The script assumes that the configuration files are located at "/etc/asterisk" and uses the following files:
    - "sip_additional.conf"
    - "iax.conf"

  Your server may have a different configuration regarding the location and the name of the files, the script can be easily adapted to match your configuration.

- DND, the script allows to control the server side DND (using the phpagi) and displays the DND status as a message on the idle screen. It also sets the LED attached to the DND key.

- Call Forward, the script allows to control the server side immediate Call forward (using the phpagi) and displays the Call forward status as a message on the idle screen. It also sets the LED attached to the Call Forward key.

- Register, the script synchronizes the server status for DND and Call Forward on the phone. The idle screen messages and the LED status are lost after a reboot, so typically this script should be called using the action register uri to update the display after a reboot.

Note: the scripts are php files, they should not be located under the "cgi-bin" directory or they would not be interpreted properly by the HTTP server.

Configuration

For this configuration, it is assumed that the scripts are located in the same directory at http://myserver.com.

- 55i/57i/57iCT

```
# Softkey 1
softkey1 type: xml
softkey1 label: Directory
softkey1 value: http://myserver.com/directory.php

# Softkey 2
softkey2 type: xml
softkey2 label: DND
softkey2 value:
http://myserver.com/dnd.php?user=$$SIPUSERNAME$$&key=softkey2
```

```
# Softkey 3
softkey3 type: xml
softkey3 label: Call Fwd
softkey3 value:
http://myserver.com/cfwd.php?user=$$SIPUSERNAME$$&key=softkey3

# Register uri
action uri registered:
http://myserver.com/register.php?user=$$SIPUSERNAME$$&dndkey=softke
y2&cfkey=softkey3
```

- 53i

```
# Key 3
prgkey1 type: xml
prgkey1 value: http://myserver.com/directory.php

# Key 4
prgkey2 type: xml
prgkey2 value:
http://myserver.com/dnd.php?user=$$SIPUSERNAME$$&key=prgkey4

# Key 5
prgkey3 type: xml
prgkey3 value:
http://myserver.com/cfwd.php?user=$$SIPUSERNAME$$&key=prgkey5

# Register uri
action uri registered:
http://myserver.com/register.php?user=$$SIPUSERNAME$$&dndkey=prgkey
4&cfkey=prgkey5
```

Note: the source code is not supported by Aastra, it is provided as an example.

15.2 Directory

The following example show how to create a simple directory for the Asterisk users configured on your Asterisk server.

```
<?
#########################################################################
# Asterisk Directory for Aastra SIP Phones R2.0.2 or better
#
# php source code
# Provided by Aastra Telecom Ltd 2007
#
# Parses
#    - SIP users
#    - IAX users
#
# Warning
#    Location of Asterisk config files is set to "/etc/asterisk"
#########################################################################

function Aastra_decode_HTTP_header()
```

```
{
$user_agent=$_SERVER["HTTP_USER_AGENT"];
if(stristr($user_agent,"Aastra"))
        {
        $value=preg_split("/ MAC:/",$user_agent);
        $fin=preg_split("/ /",$value[1]);
        $value[1]=preg_replace("/\-/","",$fin[0]);
        $value[2]=preg_replace("/V:/","",$fin[1]);
        }
else
        {
        $value[0]="MSIE";
        $value[1]="NA";
        $value[2]="NA";
        }

return($value);
}

# Location of asterisk config files
$location = "/etc/asterisk/";

# Global Variables
$Server = "http://". $_SERVER['SERVER_ADDR'].$_SERVER['SCRIPT_NAME'];

# Init number of records
$index=0;

# Parse sip.conf
$sip_array=parse_ini_file($location."sip_additional.conf", true);
while ($v=current($sip_array))
        {
        if((isset($v['callerid'])) and (key($sip_array)!="register"))
                {
                $temp=$v['callerid'];
                $len=strlen($temp);
                $callerid=substr($temp,0,$len-(strlen(strrchr($temp, '<'))));
                $directory[] = "<Prompt>".
$callerid."</Prompt>\n"."<URI>Dial:".key($sip_array)."</URI>\n"."<Selection>".key($
sip_array)."</Selection>\n"."<Dial>".key($sip_array)."</Dial>\n";
                $index++;
                }
        next($sip_array);
        }

# Parse iax.conf
$iax_array=parse_ini_file($location."iax.conf", true);
while($v=current($iax_array))
        {
        if(isset($v['name']))
                {

        $directory[]="<Prompt>".$v['name']."</Prompt>\n"."<URI>Dial:".key($iax_array
)."</URI>\n"."<Dial>".key($iax_array)."</Dial>\n";
                $index++;
                }
        next($iax_array);
        }

# Sort Directory
sort($directory);

# Get header info
$header=Aastra_decode_HTTP_header();
switch($header[0])
        {
        case 'Aastra53i':
                $MaxLines=13;
                break;
        default:
                $MaxLines=15;
```

```
                break;
        }

# Retrieve last page
$last=intval($index/$MaxLines);
if(($index-$last*$MaxLines) != 0) $last++;

# Retrieve current page
$page=$_GET['page'];
if (empty($page)) $page=1;

# Display Page
$output ="<AastraIPPhoneTextMenu destroyOnExit=\"yes\">";
$output .= "<Title>Directory ($page/$last)</Title>\n";
$index=1;
foreach ($directory as $v)
        {
        if(($index>=(($page-1)*$MaxLines+1)) and ($index<=$page*$MaxLines))
                {
                $output .= "<MenuItem>\n";
                $output .= $v;
                $output .= "</MenuItem>\n";
                }
        $index++;
        }

# Depending on the phone
switch($header[0])
        {
        case 'Aastra53i':
                # Previous button as a menu
                if($page!=1)
                        {
                        $previous=$page-1;
                        $output .= "<MenuItem>\n";
                        $output .=
"<Prompt>Previous</Prompt>\n"."<URI>".$Server."?page=".$previous."</URI>\n";
                        $output .= "</MenuItem>\n";
                        }
                # Next button as a menu
                if($page!=$last)
                        {
                        $next=$page+1;
                        $output .= "<MenuItem>\n";
                        $output .= "<Prompt>Next</Prompt>\n";
                        $output .= "<URI>".$Server."?page=".$next."</URI>\n";
                        $output .= "</MenuItem>\n";
                        }
                break;
        default:
                # Dial button
                $output .= "<SoftKey index=\"1\">\n";
                $output .= "<Label>Dial</Label>\n";
                $output .= "<URI>SoftKey:Select</URI>\n";
                $output .= "</SoftKey>\n";

                # Next button
                if($page!=$last)
                        {
                        $next=$page+1;
                        $output .= "<SoftKey index=\"5\">\n";
                        $output .= "<Label>Next</Label>\n";
                        $output .= "<URI>$Server?page=$next</URI>\n";
                        $output .= "</SoftKey>\n";
                        }

                # Previous button
                if($page!=1)
                        {
                        $previous=$page-1;
                        $output .= "<SoftKey index=\"2\">\n";
```

```
                        $output .= "<Label>Previous</Label>\n";
                        $output .= "<URI>$Server?page=$previous</URI>\n";
                        $output .= "</SoftKey>\n";
                        }

                # Exit Button
                $output .= "<SoftKey index=\"6\">\n";
                $output .= "<Label>Exit</Label>\n";
                $output .= "<URI>SoftKey:Exit</URI>\n";
                $output .= "</SoftKey>\n";
                break;
        }

# End of the object
$output .= "</AastraIPPhoneTextMenu>\n";

# HTTP header and output
header("Content-Type: text/xml");
header("Content-Length: ".strlen($output));
echo $output;
?>
```

15.3 DND

This example requires the phpagi includes.

The directory number must be passed as an argument to the script as well as the key supporting the DND feature.

`script.php?user=XXXX&key=YY` where

- XXXX is the directory number of the user.
- YY is keyID calling the XML script

The system variable $$SIPUSERNAME$$ can be used to call the script from a softkey.

```
<?
########################################################################
# Asterisk DND for Aastra SIP Phones R2.0.2 or better
#
# php source code
# Provided by Aastra Telecom Ltd 2006
#
# @user=extension
# @action=change or check
# @key identification of the key used for the function
########################################################################

include (dirname(__FILE__)."/phpagi/misc.php");
include (dirname(__FILE__)."/phpagi/phpagi-asmanager.php");

########################################################################
# Aastra_decode_HTTP_header
#
# Returns an array
#   0 Phone Type
#   1 Phone MAC Address
#   2 Phone firmware version
########################################################################

function Aastra_decode_HTTP_header()
{
$user_agent=$_SERVER["HTTP_USER_AGENT"];
```

```
if(stristr($user_agent,"Aastra"))
        {
        $value=preg_split("/ MAC:/",$user_agent);
        $fin=preg_split("/ /",$value[1]);
        $value[1]=preg_replace("/\-/","",$fin[0]);
        $value[2]=preg_replace("/V:/","",$fin[1]);
        }
else
        {
        $value[0]="MSIE";
        $value[1]="NA";
        $value[2]="NA";
        }

return($value);
}

function Aastra_manage_dnd($user,$action)
{
# Connect to AGI
$as = new AGI_AsteriskManager();
$res = $as->connect();

#DND GET
$res = $as->Command('database get DND '.$user);
$line=split("\n", $res['data']);
$value=split(" ", $line[0]);
if($value[1]=="YES") $dnd=1;
if($dnd==0)
        {
        $value=split(" ", $line[1]);
        if($value[1]=="YES") $dnd=1;
        }

if($action=='change')
        {
        # change DND status
        if($dnd==0)
                {
                $res = $as->Command('database put DND '.$user.' YES');
                $dnd=1;
                }
        else
                {
                $res = $as->Command('database del DND '.$user);
                $dnd=0;
                }
        }

# Disconnect properly
$as->disconnect();

return($dnd);
}

########################################################################
# Global parameters
########################################################################
$Server = "http://". $_SERVER['SERVER_ADDR'].$_SERVER['SCRIPT_NAME'];
$dnd=0;

# Retrieve parameters
$user=$_GET['user'];
$action=$_GET['action'];
$status=$_GET['status'];
$key=$_GET['key'];

# Force default action
if($action=="") $action="change";

# Get header info
```

```
$header=Aastra_decode_HTTP_header();

# Setup header type
header("Content-Type: text/xml");

# Depending on action
switch($action)
	{
	case 'display':
		$output = "<AastraIPPhoneFormattedTextScreen destroyOnExit=\"yes\"
Timeout=\"2\">\n";
		$output .= "<Line/>\n";
		$output .= "<Line/>\n";
		if ($status==1) $output .= "<Line Size=\"double\"
Align=\"center\">DND activated</Line>\n";
		else $output .= "<Line Size=\"double\" Align=\"center\">DND
deactivated</Line>\n";
		$output .= "<SoftKey index=\"6\">\n";
		$output .= "<Label> </Label>\n";
		$output .= "<URI>SoftKey:Exit</URI>\n";
		$output .= "</SoftKey>\n";
		$output .= "</AastraIPPhoneFormattedTextScreen>\n";
		break;

	case 'msg':
		$output = "<AastraIPPhoneStatus>\n";
		$output .= "<Session>CFDND</Session>\n";
		if ($status==1) $output .= "<Message index=\"0\">DND
activated</Message>\n";
		else $output .= "<Message index=\"0\"></Message>\n";
		$output .= "</AastraIPPhoneStatus>\n";
		break;

	case 'change':
	case 'check':
		$dnd=Aastra_manage_dnd($user,$action);
		$output = "<AastraIPPhoneExecute>\n";
		$output .= "<ExecuteItem
URI=\"".$Server."?action=msg&amp;status=".$dnd."\"/>\n";
		if ($dnd==1) $output .= "<ExecuteItem URI=\"Led: ".$key."=on\"/>\n";
		else $output .= "<ExecuteItem URI=\"Led: ".$key."=off\"/>\n";
		switch($header[0])
			{
			case "Aastra53i":
				break;

			default:
				if($action=='change') $output .= "<ExecuteItem
URI=\"".$Server."?action=display&amp;status=".$dnd."\"/>\n";
				break;
			}
		$output .= "</AastraIPPhoneExecute>\n";
		break;
	}

header("Content-Length: ".strlen($output));
echo $output;
?>
```

15.4 Call Forward

This example requires the phpagi includes.

The directory number must be passed as an argument to the script.

`script.php?user=XXXX&key=YY` where

- XXXX is the directory number of the user.
- YY is keyID calling the XML script

The system variable $$SIPUSERNAME$$ can be used to call the script from a softkey.

```
<?
########################################################################
# Asterisk Call Forward for Aastra SIP Phones R2.1.0 or better
#
# php source code
# Provided by Aastra Telecom Ltd 2006
#
# @user extension
# @action '' or check
# @key identification of the key used for the function
########################################################################

include (dirname(__FILE__)."/phpagi/misc.php");
include (dirname(__FILE__)."/phpagi/phpagi-asmanager.php");

########################################################################
# Aastra_decode_HTTP_header
#
# Returns an array
#   0 Phone Type
#   1 Phone MAC Address
#   2 Phone firmware version
########################################################################

function Aastra_decode_HTTP_header()
{
$user_agent=$_SERVER["HTTP_USER_AGENT"];
if(stristr($user_agent,"Aastra"))
        {
        $value=preg_split("/ MAC:/",$user_agent);
        $fin=preg_split("/ /",$value[1]);
        $value[1]=preg_replace("/\-/","",$fin[0]);
        $value[2]=preg_replace("/V:/","",$fin[1]);
        }
else
        {
        $value[0]="MSIE";
        $value[1]="NA";
        $value[2]="NA";
        }

return($value);
}

function Aastra_manage_cf($user,$action,$value)
{
# Connect to AGI
$cf="";
$as = new AGI_AsteriskManager();
$res = $as->connect();

# Depending on action
switch($action)
        {
        case 'cancel':
                $res = $as->Command('database del CF '.$user);
                break;
        case 'set':
                $res = $as->Command('database put CF '.$user.' '.$value);
                $cf=$value;
                break;
        default:
                #GET CFWD
                $res = $as->Command('database get CF '.$user);
```

```
                $line=split("\n", $res['data']);
                $data=split(" ", $line[0]);
                if($data[0]=="Value:") $cf=$data[1];
                if($cf=="")
                        {
                        $data=split(" ", $line[1]);
                        if($data[0]=="Value:") $cf=$data[1];
                        }
                break;
        }

# Disconnect properly
$as->disconnect();

return($cf);
}

########################################################################
# Global parameters
$Server = "http://". $_SERVER['SERVER_ADDR'].$_SERVER['SCRIPT_NAME'];

# Retrieve parameters
$user=$_GET['user'];
$action=$_GET['action'];
$value=$_GET['value'];
$key=$_GET['key'];

# Get header info
$header=Aastra_decode_HTTP_header();

# Depending on action
switch($action)
        {
        case 'cancel':
        case 'set':
        case 'check':
                $cf=Aastra_manage_cf($user,$action,$value);
                $output = "<AastraIPPhoneExecute triggerDestroyOnExit=\"yes\">\n";
                $output .= "<ExecuteItem
URI=\"".$Server."?action=msg&amp;user=".$user."\"/>\n";
                if($cf=='') $output .= "<ExecuteItem URI=\"Led: ".$key."=off\"/>\n";
                else $output .= "<ExecuteItem URI=\"Led: ".$key."=on\"/>\n";
                if($action!='check')
                        {
                        switch($header[0])
                                {
                                case "Aastra53i":
                                        break;
                                default:
                                        $output .= "<ExecuteItem
URI=\"".$Server."?user=".$user."&amp;key=".$key."\"/>\n";
                                        break;
                                }
                        }
                $output .= "</AastraIPPhoneExecute>\n";
                break;

        case 'change':
                $cf=Aastra_manage_cf($user,$action,$value);
                $output = "<AastraIPPhoneInputScreen type=\"number\"
destroyOnExit=\"yes\">\n";
                $output .= "<Title>Call Forward</Title>\n";
                $output .= "<Prompt>Enter destination</Prompt>\n";
                $output .=
"<URL>".$Server."?user=$user&amp;action=set&amp;key=$key</URL>\n";
                $output .= "<Parameter>value</Parameter>\n";
                $output .= "<Default>$cf</Default>\n";
                $output .= "</AastraIPPhoneInputScreen>\n";
                break;

        case 'msg':
```

```
                $cf=Aastra_manage_cf($user,$action,$value);
                $output = "<AastraIPPhoneStatus>\n";
                $output .= "<Session>CFDND</Session>\n";
                if ($cf=="") $output .= "<Message index=\"1\"></Message>\n";
                else $output .= "<Message index=\"1\">CFWD activated</Message>\n";
                $output .= "</AastraIPPhoneStatus>\n";
                break;

        default:
                $cf=Aastra_manage_cf($user,$action,$value);
                switch($header[0])
                        {
                        case "Aastra53i":
                                $output = "<AastraIPPhoneTextMenu
destroyOnExit=\"yes\">\n";
                                if($cf=="") $output .= "<Title>CFWD
deactivated</Title>\n";
                                else $output .= "<Title>CFWD set
(".$cf.")</Title>\n";
                                if($cf!="")
                                        {
                                        $output .= "<MenuItem>\n";
                                        $output .= "<Prompt>Cancel</Prompt>\n";
                                        $output .=
"<URI>$Server/cfwd.php?action=cancel&amp;user=$user&amp;key=$key</URI>\n";
                                        $output .= "</MenuItem>\n";
                                        }
                                $output .= "<MenuItem>\n";
                                $output .= "<Prompt>Change</Prompt>\n";
                                $output .=
"<URI>$Server/cfwd.php?action=change&amp;user=$user&amp;key=$key</URI>\n";
                                $output .= "</MenuItem>\n";
                                $output .= "</AastraIPPhoneTextMenu>\n";
                                break;

                        default:
                                $output = "<AastraIPPhoneTextScreen
destroyOnExit=\"yes\">\n";
                                $output .= "<Title>Call Forward for $user</Title>\n";
                                if ($cf=="") $output .= "<Text>Call Forward is
currently deactivated.</Text>\n";
                                else $output .= "<Text>Call Forward is currently set
to $cf.</Text>\n";
                                $output .= "<SoftKey index=\"1\">\n";
                                $output .= "<Label>Change</Label>\n";
                                $output .=
"<URI>$Server/cfwd.php?action=change&amp;user=$user&amp;key=$key</URI>\n";
                                $output .= "</SoftKey>\n";
                                if($cf!="")
                                        {
                                        $output .= "<SoftKey index=\"2\">\n";
                                        $output .= "<Label>Cancel</Label>\n";
                                        $output .=
"<URI>$Server/cfwd.php?action=cancel&amp;user=$user&amp;key=$key</URI>\n";
                                        $output .= "</SoftKey>\n";
                                        }
                                $output .= "<SoftKey index=\"6\">\n";
                                $output .= "<Label>Done</Label>\n";
                                $output .= "<URI>SoftKey:Exit</URI>\n";
                                $output .= "</SoftKey>\n";
                                $output .= "</AastraIPPhoneTextScreen>\n";
                                break;
                        }
                break;
        }

header("Content-Type: text/xml");
header("Content-Length: ".strlen($output));
echo $output;
exit;
?>
```

15.5 Register

The directory number must be passed as an argument to the script as well as the key supporting the DND feature (on a 55i/57i and 57i CT only).

This script assumes

- the DND script is called “dnd.php”,
- the Call Forward script is call “cfwd.php”,
- all the scripts are in the same directory

`script.php?user=XXXX&dndkey=YY&cfkey=ZZ` where

- XXXX is the directory number of the user.
- YY is the ID of the key calling the DND XML script
- ZZ is the ID of the key calling the CFWD XML script

```
<?
###############################################################################
# Asterisk Register - Update DND and CFWD status
#
# Aastra SIP Phones 2.0.2 or better
#
# Copyright 2006 Aastra Telecom Ltd
#
# script.php?user=XXX&dndkey=YYY&cfkey=ZZZ
# XXX is the extension of the phone on the platform
# YYY is the number of the dynamic dnd key
# ZZZ is the number of the dynamic cfwd key
#
###############################################################################

#########################################################################
# Global parameters
$Server = "http://". $_SERVER['SERVER_ADDR'].$_SERVER['SCRIPT_NAME'];
$Server = substr($Server,0,(strlen($Server)-strlen(strrchr($Server, "/"))+1));

# Retrieve parameters
$user=$_GET['user'];
$dndkey=$_GET['dndkey'];
$cfkey=$_GET['cfkey'];

# Update DND and CFWD
$output = "<AastraIPPhoneExecute>\n";
$output .= "<ExecuteItem
URI=\"".$Server."dnd.php?action=check&amp;user=".$user."&amp;key=".$dndkey."\"/>\n"
;
$output .= "<ExecuteItem
URI=\"".$Server."cfwd.php?action=check&amp;user=".$user."&amp;key=".$cfkey."\"/>\n"
;
$output .= "</AastraIPPhoneExecute>\n";

# Return object
header("Content-Type: text/xml");
header("Content-Length: ".strlen($output));
echo $output;
exit;
?>
```