



OpenStage 60/80 - XML Applications

We are now **Unify**.

Formerly known as Siemens Enterprise Communications, Unify continues to be one of the world's largest communications software and services firms.

This document references our previous company name; all other content is still valid and correct. For further information, please visit unify.com.

Documentation

OpenStage 60/80 - XML Applications

Developer's Guide

A31003-S2000-R100-9-7620



Communication for the open minded

Siemens Enterprise Communications
www.siemens-enterprise.com

SIEMENS

Our Quality and Environmental Management Systems are implemented according to the requirements of the ISO9001 and ISO14001 standard certified by an external certification company.

Copyright © Siemens Enterprise Communications GmbH & Co. KG 2004
Hofmannstr. 51, 80200 München

Siemens Enterprise Communications GmbH & Co. KG is a Trademark Licensee of Siemens AG

Reference No.: A31003-S2000-R100-9-7620

The information provided in this document contains merely general descriptions or characteristics of performance which in case of actual use do not always apply as described or which may change as a result of further development of the products. An obligation to provide the respective characteristics shall only exist if expressly agreed in the terms of contract. Availability and technical specifications are subject to change without notice. OpenScape, OpenStage and HiPath are registered trademarks of Siemens Enterprise Communications GmbH & Co. KG. All other company, brand, product and service names are trademarks or registered trademarks of their respective holders.

Communication for the open minded

Siemens Enterprise Communications
www.siemens-enterprise.com

1 Introduction	3
1.1 Overview	3
1.2 Prerequisites	4
2 Operating Mode and Features	5
2.1 Infrastructure	5
2.2 Basic Operating Mode	6
2.3 Push Mode	7
2.4 Cache	7
3 Setup, Start and Use	9
3.1 Configuring an Application on the Phone	9
3.2 Setting a Proxy	17
3.3 Reconfigure and Remove an Application	19
3.4 Starting the Application	22
3.5 Mobile Users And Privacy	24
3.6 User Interface	25
4 XML Structure and Encoding	31
4.1 Functionality and Structure of an XML Application	31
4.2 Encoding/Internationalization	33
5 XML Object Reference	34
5.1 Phone	34
5.2 Display	35
5.3 Screen	37
5.4 Action	39
5.5 Command	43
5.6 Direct Keypad Input	49
5.7 Alert	57
5.8 List	62
5.9 Text Box	71
5.10 Ticker	75
5.11 Hidden	77
5.12 Form	79
5.13 Form/Item	82
5.14 Form/String Item	86
5.15 Form/Image Item	89
5.16 Form/Spacer	92
5.17 Form/Text Field	95
5.18 Form/Choice Group	99
5.19 Form/Date Field	104
5.20 Form/Button	109
5.21 Form/Gauge	112
5.22 Image	117
5.23 Player	119
5.24 Phone Number	126

6 Push Capability	128
6.1 Push Types	128
6.2 The Push Request	132
7 Using Additional Phone Keys	135
7.1 Overview	135
7.2 Configuring the Phone	135
7.3 Phone Request	137
7.4 Creating the Push Request for LED Control (V2R2)	138
7.5 XML Document for LED Control	139
8 Appendix	141
8.1 Glossary	141
8.2 XML Schema for XML Applications	142
8.3 XML Schema for the Send URL Feature	170

1 Introduction

1.1 Overview

With their graphical user interface, OpenStage 60/80 models allow for creating applications for a multitude of purposes.

As XML and HTTP serve as client/server interface, the technologies commonly used in web applications can be used: Java Servlets, JSP, PHP, CGI etc., delivered by servers such as Tomcat, Apache, Microsoft IIS.

The Push feature enables the server-side application to send data to the phone in an unsolicited way.

The XML application interface is available in the SIP firmware (from V1R3.x) or the HFA firmware (from V2R0.x) for OpenStage 60/80 phones.

This document describes the XML interface and the error handling in case the XML data is invalid. Please note that the error handling is not visible to the user. If, for instance, some elements of an object are missing, the remaining objects are displayed anyhow. It is recommended to validate the XML documents using the XML schema provided here (see Section 8.2, "XML Schema for XML Applications").

This document is aimed at Siemens and Third Party developers and covers the following areas:

- Operating Mode and Features
- Setup, Start and Use
- XML Object Reference
- Push Capability
- Appendix

1.2 Prerequisites

For developing applications for OpenStage phones, the following components and tools are required resp. recommended:

1. OpenStage 60/80 SIP/HFA phone with appropriate firmware version (V1R3.x or higher).
2. Infrastructure:
 - Connection to an IP network.
 - Web browser to set up the application on the phone using the phone's Web Based Management. Alternatively, the phone's local admin menu can be used.
 - Web server for sending and receiving XML data, like Tomcat, Apache, Microsoft IIS. Please note that the MIME type for the XML documents has to be set to `text/xml` on the server.
 - Application platform running on the server, for instance Java Servlet, ASP, CGI, PHP.
3. XML Editor:

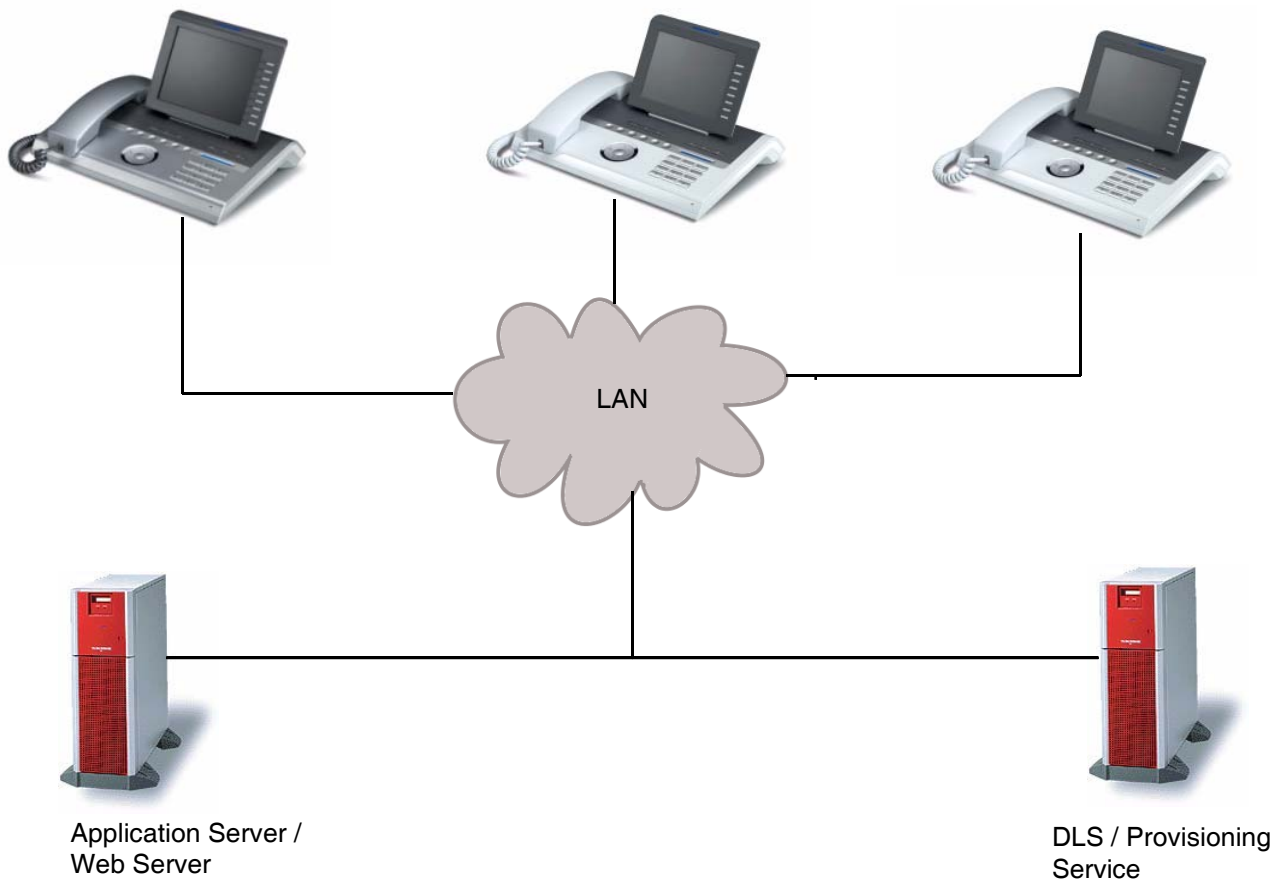
A validating XML editor is recommended, like Oxygen, XMLWriter, EditiX, XMLSpy.
4. For developing Java Servlets: An applicable development environment, for instance Eclipse, NetBeans, IBM Websphere Studio.

2 Operating Mode and Features

2.1 Infrastructure

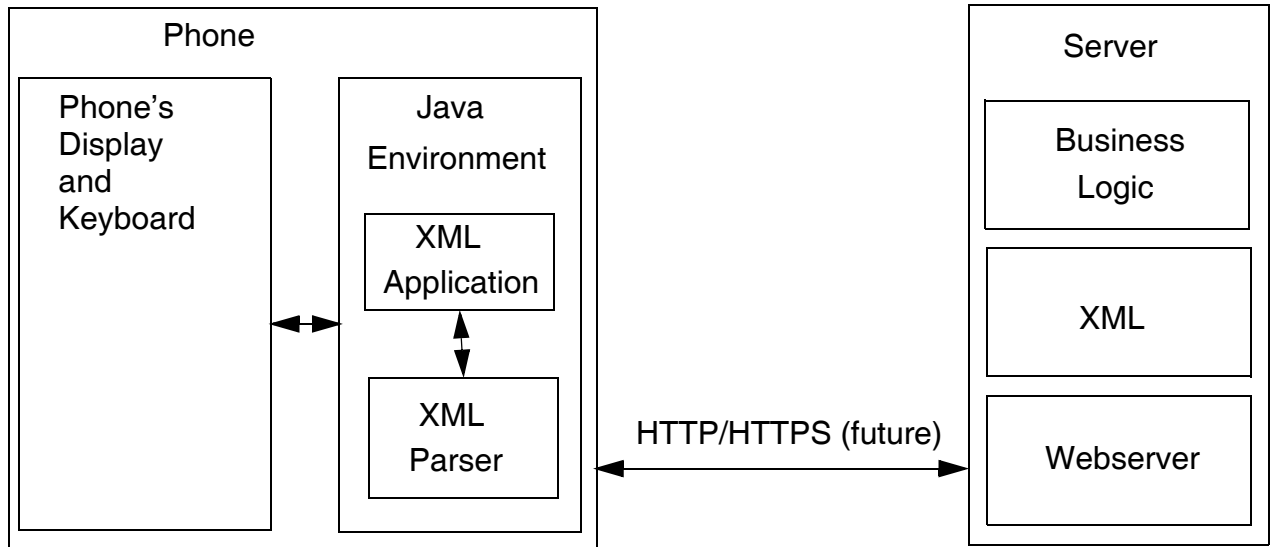
The phones are connected to a LAN. The application/web server can be connected to the same LAN, or alternatively be reachable via WAN/internet. Optionally, the DLS (Deployment Service) or the phone's provisioning interface can be used to configure XML applications on the phone. For details, please refer to the OpenScape Deployment Service Administration and Installation Manual and to the OpenStage Provisioning Service Developer's Guide.

The following diagram shows the communication between the components:



2.2 Basic Operating Mode

The following diagram shows the basic components involved in the operation:



The operation can be described as follows:

1. The administrator configures one or more applications on the OpenStage phone.
2. The user selects a program from the list of available applications and starts it.
3. The phone sends an HTTP request to the URL of the server-side application.



The HTTP/HTTPS (future) GET request sent by the phone on application startup contains the phone's IP address and call number as well as the device type.

Example:

```
GET /ipp/example?ipaddress=192.168.1.244&
phonenum=3338&devicetype=OpenStage
```

4. The server-side program generates an XML document which is valid according to the XML schema (see Section 8.2, "XML Schema for XML Applications"). The web server delivers the XML document to the phone over an HTTP connection.
5. In the phone, the XML document is parsed and displayed on the graphic display.
6. The user enters commands and data using the phone's TouchGuide and keypad.
7. Listeners in the phone software recognize the commands and data entered by the user.
8. The phone transmits the user entries to the server-side program in the form of key/value pairs.

2.3 Push Mode

The push mode enables the server-side program to control and start a phone-side XML application.

The basic operation of the push capability can be described as follows: A program running on a server calls a special URL of the phone, which is based on the phone's IP address. Through this, a specific XML application on the phone is triggered to load an XML document from a URL given in the push request. If the specific XML application is not already running on the phone, it may be started automatically.

2.4 Cache

The cache stores up to 20 screen contents of any type (form, list, text box, alert, or player) on the phone. When the cache is full, all of the entries in the cache are shifted one position to the left and the new screen is added to the end. However, there is an exception to this rule: If the cache consists of only one full screen and the rest alerts, the shifting of the entries in the cache will retain the full screen so that there is a screen to go back to when all of the alerts have been dismissed.

Any screen in the cache may have links to other screens in the cache. They are linked by BACK and SCREEN commands (see Section 5.5, "Command"). An alert or player is removed from the cache when there are no links to it from any other screens.

When new screens are received from the remote server, the new screen may:

- Reuse the existing display items of an existing screen in the cache, or
- overwrite an existing screen in the cache, or
- create a new screen in the cache.

When a screen is overwritten, the existing display items are deleted and a new set of display items are created for the new screen.

The following table details the list reuse and screen caching rules when a new screen is received:

Screen Type	Has BACK Command		No BACK Command	
	Screen ID already exists in cache	Screen ID does not exist in cache	Screen ID already exists in cache	Screen ID does not exist in cache
Alert	When alerts are pushed, a new alert is created. Otherwise, the existing alert is overwritten.	A new alert is created.	When alerts are pushed, a new alert is created. Otherwise, the existing alert is overwritten.	A new alert is created.
Form	An existing form is overwritten.	A new form is created.	An existing form is overwritten.	A new form is created.
List	A new list is created.	A new list is created.	The existing list is reused.	A new list is created.
Player	The existing player is overwritten.	A new player is created.	The existing player is overwritten.	A new player is created.
Text Box	The existing text box is overwritten.	A new text box is created.	The existing text box is overwritten.	A new text box is created.

Table 2-1

3 Setup, Start and Use

3.1 Configuring an Application on the Phone

For setting up an XML Application, the WBM (Web Based Manager), the local admin menu, the DLS (Deployment Service), or the phone's provisioning interface can be used. For details, please refer to the OpenScape Deployment Service Administration and Installation Manual and to the OpenStage Provisioning Service Developer's Guide.



A maximum number of 20 XML applications can be configured on OpenStage 60/80 phones.






Please ensure that the following prerequisites are fulfilled:

- The phone has firmware version SIP V1R3.x or higher, or HFA V2R0.x or higher
- The terminal number is set (for details, see the OpenStage Administration Guide).
- The phone has intranet/internet access, directly or via proxy. For details about setting a proxy, see Section 3.2, "Setting a Proxy", or the OpenStage Administration Guide.

3.1.1 Application Types

From the user's standpoint, there is a choice of several types of XML applications, which mainly differ in the way they are started and stopped:

- Regular XML applications are started by navigating to the applications menu using the ☰ key, or by pressing a programmable key (see Section 3.4, "Starting the Application"). They can be stopped via the applications menu. Via the web management, regular XML applications are configured under **Applications > XML applications > Add application**.
- Xpressions is a special Unified Communications application which also uses the XML interface. Thus, the configuration is just the same as with other XML applications, except a few parameters, which are pre-configured. For details, please refer to the relevant Xpressions documentation. When configured on the phone, a press on the messages mode key ☑ will invoke this application. Via the web management, an application Xpressions is configured under **Applications > XML applications > Xpressions**.
- A messages application is configured like a regular application. It is started via the messages mode key ☑, thus enabling the deployment of an alternative voicemail server. From firmware version V2R1 onwards, the XML application can control the white LED which indicates new messages. Via the web management, a messages application is configured under **Applications > XML applications > Add messages application**.

- A phonebook application is configured like a regular application. It is started via the phonebook mode key , thus enabling the deployment of a remote phonebook in place of the personal (local) or corporate (LDAP) phonebook. Via the web management, a phonebook application is configured under **Applications > XML applications > Add phonebook application**.
- A call log application is configured like a regular application. It is started via the call log mode key , thus enabling the deployment of a remote application that handles call history. From firmware version V2R1 onwards, the XML application can control the white LED which indicates missed calls. Via the web management, a call log application is configured under **Applications > XML applications > Add call log application**.
- A help application is configured like a regular application. It is started via the help mode key , thus enabling the deployment of a remote help. Via the web management, a help application is configured via **Applications > XML applications > Add help application**.

3.1.2 Required Data

An XML application is set up by providing the following parameters:

- **Display name:** Name of the application which is displayed on the application tab of the phone's display. The following criteria must be satisfied:
 - It must be unique on the phone.
 - It cannot contain the '^' character.
 - It cannot be empty.
 - Its length cannot not exceed 20 characters.
- **Application name:** Used internally to identify the XML application running on the phone. The application name must meet the following criteria:
 - It must be unique on the phone.
 - It cannot contain non-alphanumeric characters, spaces for instance.
 - The first character must be a letter.
 - It must not be empty.
 - Its length must not exceed 20 characters.
- **Protocol:** The protocol used for communication to the server. Currently, only HTTP is supported. Support of HTTPS is planned for the future.
- **Server address:** The IP address or domain/host name of the server that provides the application or the XML document.
Examples: 192.168.1.133, backoffice.intranet

Setup, Start and Use

Configuring an Application on the Phone

- **Server port number:** The number of the port used by the server to provide the XML documents for the application.
Examples: 80 (Apache default port), 8080 (Tomcat default port).
- **Program name:** The relative path to the remote program, e. g. to the servlet or to the first XML page. With static start pages, for instance, the relative path refers to the root directory for documents on the web server. For instance, if an XML document is saved in:
C:\Program Files\Apache Group\Apache\htdocs\ipp\ippTest.xml
the entry is:
ipp/ippTest.xml.
The program name cannot be longer than 100 characters.
- **Auto start** (V2R1 onwards) determines whether the application is started automatically on phone startup or, with V2R2 onwards, on mobile user logon. Please note that, for being started on logon, the application must be part of the mobile user's profile. When activated, the application will be ready without delay as soon as the user presses the corresponding start key or navigates to the application in the application menu.
- **Use proxy:** Enables an HTTP proxy for communication with the server, if desired. If disabled, a direct connection is used. For details, see Section 3.2, "Setting a Proxy".
- **XML trace enabled:** Enables or disables the debugging of the XML application back to the designated program specified by **Debug program name**. When enabled, trace information about the XML elements and key internal objects is sent to the remote debug program.
- **Debug program name:** The relative path to a special program on the same server as the program specified by **Program name**. This program must be able to receive the debug information sent by the phone as HTTP POST requests with `Content-Type` set to `application/x-www-form-urlencoded`.
- **Number of tabs:** Required if the application has internal tabs. Specifies the number of internal tabs for this application. The maximum value is 3.



For an XML application with a number of tabs > 0, one of the entries between **Tab 1 Application Name** and **Tab 3 Application Name** must be set to the same value as the **Application Name** that it is associated with. This tab will have the focus when the XML application is started.

- **All tabs start** (V2R1 onwards) determines whether all tabs of the application are started automatically when the application is started.
- **Tab 1...3 Display Name:** Required if the application has internal tabs. Provides the label text for the specified tab.

- **Tab 1...3 Application Name:** Required if the application has internal tabs. Provides a unique name for the specified tab. The remote program will use this name to provide the tab with specific content. For more information, please refer to Section 3.4, "Application Suite Using Predefined Tabs" and Section 3.4, "Application Suite Using Tabs On Demand".
- **Restart after change:** If checked, a running XML application is automatically restarted after it has been modified. After the XML application has restarted, this option is automatically unchecked.

Setup, Start and Use

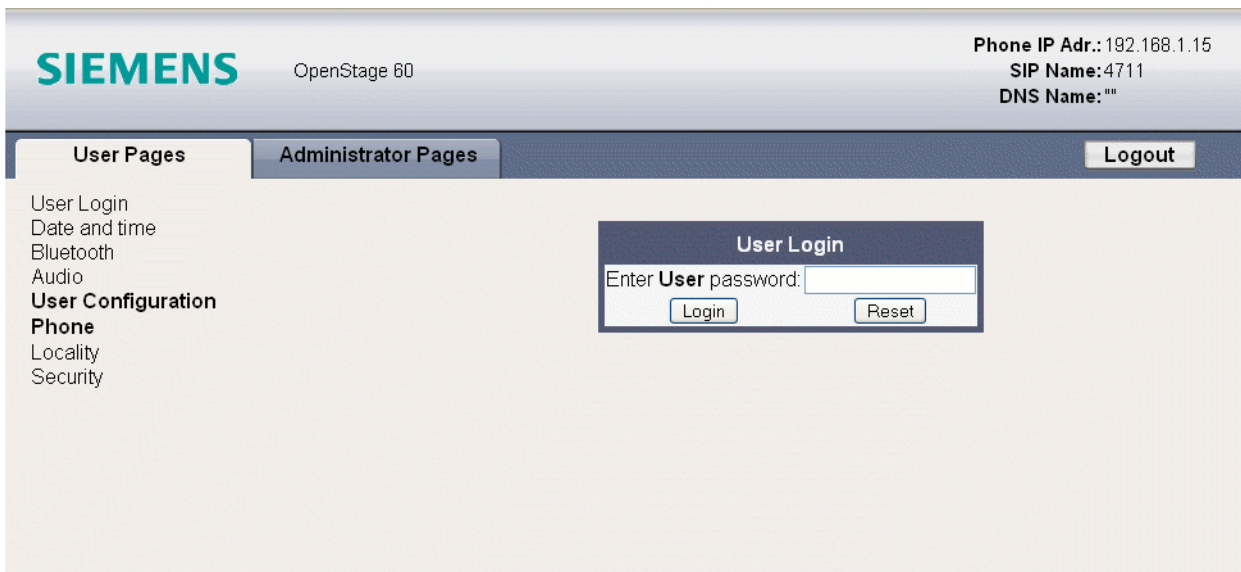
Configuring an Application on the Phone

3.1.3 Using the Web Based Management (WBM)

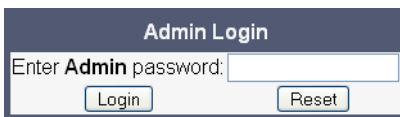
1. Open your web browser (MS Internet Explorer or Firefox) and enter the appropriate URL. Example: `https://192.168.1.15` or `https://myphone.phones` (firmware V2)

For configuring the phone's DNS name, which is possible with firmware V2, please refer to the OpenStage Administration Manual.

If the browser displays a certificate notification, confirm it. The start page of the web based management appears. In the upper right corner, the phone number, the phone's IP address, and the DNS name assigned to the phone are displayed. The left column contains the user menu tree.



2. Click on the **Administrator Pages** tab. In the dialog box, enter the admin password:



3. The administrator main page opens. The navigation column to the left contains the menu tree. If you click on an item which is printed in normal style, the corresponding dialog opens in the center of the page. If you click on an item printed in bold letters, a sub-menu opens in the navigation column.

4. Navigate to **Applications > XML applications > Add application** (or **Add messages application**, or **Add phonebook application**, or **Add call log application**, or **Add help application**) and enter the required data (see Section 3.1.2, "Required Data"). When the configuration data is correct and complete, click on **Submit**.











The screenshot shows a web form titled "Add application". It contains the following fields and controls:




- Display name: text input
- Application name: text input
- HTTP Server address: text input
- HTTP Server port: text input
- Protocol: dropdown menu with "http" selected
- Program name on server: text input
- Auto start: checkbox (unchecked)
- Use proxy: dropdown menu with "Yes" selected
- XML Trace enabled: dropdown menu with "Yes" selected
- Debug program on server: text input
- Number of tabs: dropdown menu with "0" selected
- All tabs Start: checkbox (unchecked)
- Tab 1 Display Name: text input
- Tab 1 Application Name: text input
- Tab 2 Display Name: text input
- Tab 2 Application Name: text input
- Tab 3 Display Name: text input
- Tab 3 Application Name: text input
- Restart after change: checkbox (unchecked)
- Submit button
- Reset button

5. Now you can test the application. For a regular XML application, press the **☰** key on the phone to activate the **Applications** tab. If everything went well, the new application is listed and can be started by pressing **⌂**. For Xpressions, or a messages, phonebook, call log, or help application, press the corresponding mode key.
6. To modify or delete an application, repeat steps 1 to 3, and then navigate to **Modify/delete application**. Choose the desired application in the **Select application** menu and click **Modify** to change parameters, or **Delete** to delete the application.

3.1.4 Using the Local Admin Menu

The basics for accessing the phone's local admin menu are described here. For more detailed information, please consult the OpenStage 60/80 User Guide or the OpenStage Administration Manual.

1. Press the  key once or repeatedly until the **Settings** tab is active. (The  key toggles between the settings menu, the applications menu, and the applications currently running.)
2. With the TouchGuide, navigate to the **Admin** menu and press .
3. You are prompted to enter the admin password.
If the password is numeric, just type it using the number keys. On typing the first digit, an input text field will open up.
For entering non-numeric characters, press the # key once or repeatedly, depending on the desired character. The # key cycles around the input modes as follows:
(Abc) -> (abc) -> (ABC) -> (123) -> back to start.
To correct an entry, use the  key.
4. Press . The admin menu opens.
5. With the TouchGuide, navigate to the **Applications** item. To navigate, you can either run your finger around the sensor ring, or press the  or  key.
6. Press  to open the **Applications** item and select **XML**. On , **Add application** appears in the context menu. With firmware version V2R1 onwards, the other application types appear, too.
7. Press  to open the menu for configuring a new application.
8. In the **New application** menu, enter the configuration data (for details, see Section 3.1.2, "Required Data"):
 - **Display name:** Application name for display.
 - **Application name:** Internal name for the application.
 - **Server address:** IP address of server hosting the program.
 - **Server port:** Server port used by the server-side program.
 - **Protocol:** Client/server communication protocol (HTTP or, in the future, HTTPS).
 - **Program name:** Path to the server-side program.
 - **Auto start** (V2R1 onwards): Determines whether the application is started automatically on phone startup or, with V2R2, on mobile user logon.
 - **Use proxy:** Use of a direct connection or proxy for client/server communication.
 - **XML trace enabled:** Sending trace data to a debugging program.

- **Debug program name:** Path to a server-side debugging program.
 - **Number of tabs:** Number of internal tabs for this application.
 - **All tabs start** (V2R1 onwards): Relevant for application suites, that is, applications with internal tabs. Determines whether all tabs of an application suite shall be started on application start.
 - **Tab 1...3 Display Name:** Label text for the specified tab.
 - **Tab 1...3 Application Name:** Unique name for the specified tab.
 - **Restart after change:** If checked, a running XML application is restarted automatically after it has been modified.
9. When the configuration data is correct and complete, navigate to **Save & exit** and press .
 10. Press the  key to activate the **Applications** tab. If everything went well, the new application is listed here. It can be started by pressing . For Xpressions, or a messages, phone-book, call log, or help application, press the corresponding mode key.
 11. To modify or delete an application, repeat steps 1 to 6, and then select the desired application in the menu. You can change the parameters and select **Save & exit** afterwards, or delete the application by selecting **Remove & exit**.

3.2 Setting a Proxy

The HTTP data transfer between the phone and the server on which the remote program is running can be handled by an HTTP proxy, if desired. The use of a proxy can be enabled or disabled separately for each application (see Section 3.1.2, "Required Data").

3.2.1 Using the Web Interface

1. Open the web interface as described in Section 3.1.3, "Using the Web Based Management (WBM)", steps 1 to 3.
2. Navigate to **Network > IP Configuration** and enter the IP address of the proxy to be used.

IP configuration

IP address 192.168.1.12
Subnet mask 255.255.255.0
Default route 192.168.1.251
DNS domain
Primary DNS 192.168.1.105
Secondary DNS 194.25.0.53
Route 1 IP address
Route 1 gateway
Route 1 mask
Route 2 IP address
Route 2 gateway
Route 2 mask
VLAN discovery DHCP
VLAN ID
HTTP proxy

3. Press **Submit**.

4. Navigate to **Network > Port Configuration** and enter the port of the proxy to be used.

Port configuration	
SIP server	5060
SIP registrar	5060
SIP gateway	5060
SIP local	5060
Backup proxy	5060
RTP base	5010
Download server (default)	21
LDAP server	389
HTTP proxy	0
LAN port speed	Automatic
PC port speed	Automatic
PC port mode	disabled
PC port autoMDIX	<input type="checkbox"/>
<input type="button" value="Submit"/> <input type="button" value="Reset"/>	

5. Press **Submit**. The proxy is now configured.

3.2.2 Using the Local Admin Menu

1. Repeat steps 1-4 in Section 3.1.4, "Using the Local Admin Menu".
2. With the TouchGuide, navigate to the **Network** item. To navigate, you can either run your finger around the sensor ring, or press the ▲ or ▼ key.
3. Press **OK** and navigate to the **IP configuration** item. On **OK**, the submenu opens.
4. Navigate to **HTTP proxy** and press **OK**.
5. In the phone's editor, enter the IP address of the proxy. For details about using the editor, please refer to the OpenStage Administration Manual or the OpenStage 60/80 User Guide.
6. When the configuration data is correct and complete, navigate to **Save & exit** and press **OK**.

Setup, Start and Use

Reconfigure and Remove an Application

3.3 Reconfigure and Remove an Application

3.3.1 Using the Web Interface

3.3.1.1 Reconfigure

1. Open the web interface as described in Section 3.1.3, "Using the Web Based Management (WBM)", steps 1 to 3.
2. Navigate to **Applications > XML applications > Modify/Delete application** and select the application to be modified.

Modify/Delete application

Select application: testxml

Modify Delete

Settings

Display name: testxml

Application name: testxml

HTTP Server address: 192.168.1.151

HTTP Server port: 8080

Protocol: http

Program name on server: testxml/servlet

Auto start: ☒

Use proxy: No

XML Trace enabled: No

Debug program on server: ☐

Number of tabs: 0

All tabs Start: ☐

Tab 1 Display Name:

Tab 1 Application Name:

Tab 2 Display Name:

Tab 2 Application Name:

Tab 3 Display Name:

Tab 3 Application Name:

Restart after change: ☐

Mode key:

Submit Reset

- Click on **Modify**. The configuration data of the selected application is loaded into the input fields.

Modify/Delete application

Select application: testxml

Modify **Delete**

Settings

Display name	testxml
Application name	testxml
HTTP Server address	192.168.1.151
HTTP Server port	8080
Protocol	http
Program name on server	testxml/servlet
Auto start	<input checked="" type="checkbox"/>
Use proxy	No
XML Trace enabled	No
Debug program on server	<input type="checkbox"/>
Number of tabs	1
All tabs Start	<input type="checkbox"/>
Tab 1 Display Name	
Tab 1 Application Name	
Tab 2 Display Name	
Tab 2 Application Name	
Tab 3 Display Name	
Tab 3 Application Name	
Restart after change	<input type="checkbox"/>
Mode key	

Submit **Reset**

- Change the data as desired. When the configuration data is correct and complete, click on **Submit**.

Setup, Start and Use

Reconfigure and Remove an Application

3.3.1.2 Removing

1. Open the web based management as described in Section 3.1.3, "Using the Web Based Management (WBM)", steps 1 to 3.
2. Navigate to **Applications > XML applications > Modify/Delete application**.
3. In the **Select application** menu, choose the application to be deleted.



Although the contents of the **Settings** section will not change automatically when an application is selected in the menu, this selection will be valid.

Modify/Delete application

Select application testxml

Modify Delete

Settings

Display name testxml

Application name testxml

HTTP Server address 192.168.1.151

HTTP Server port 8080

Protocol http

Program name on server testxml/servlet

Auto start ☒

Use proxy No

XML Trace enabled No

Debug program on server

Number of tabs 0

All tabs Start ☐

Tab 1 Display Name

Tab 1 Application Name

Tab 2 Display Name

Tab 2 Application Name

Tab 3 Display Name

Tab 3 Application Name

Restart after change ☐

Mode key





Submit Reset

4. Press **Delete** to remove the application.





3.3.2 Using the Local Admin Menu

Here, the basics for accessing the phone's local admin menu are described. For more detailed information on using the local menu, please consult the OpenStage 60/80 User Guide or the OpenStage Administration Manual.

3.3.2.1 Reconfigure









1. Repeat steps 1-5 in Section 3.1.4, "Using the Local Admin Menu".
2. Press  to open the **Applications** item and select **XML**. On , **Add application** and the names of the previously configured applications appear in the context menu.
3. Navigate to the program that you wish to reconfigure and press .
4. In the application specific menu, enter the data required for the configuration of a program (see Section 3.1.2, "Required Data").
5. When the configuration data is correct and complete, navigate to **Save & exit** and press .

3.3.2.2 Removing

1. Repeat steps 1-5 in Section 3.1.4, "Using the Local Admin Menu".
2. Press  to open the **Applications** item and select **XML**. On , **Add application** and the names of the previously configured applications appear in the context menu.
3. Navigate to the program that you would like to delete and press .
4. The **Save & exit** option is highlighted. Open the context menu by pressing the **→** key.
5. With the TouchGuide, navigate to **Remove & exit** and press .

3.4 Starting the Application

The user can start an XML application in one of the following ways, provided that the keys are configured appropriately:

- by navigating to the applications menu using the  key,
- using a free programmable key,
- using the messages key  (messages application),
- using the phonebook key  (phonebook application),
- only with firmware version V2R1 onwards: using the forwarding key , the release key , the voice recognition key , the call log key  (call log application), or the help key  (help application).

Setup, Start and Use

Starting the Application



With firmware V2R1 onwards, an XML application can be configured so that it will be started in the background on phone start. If so, the start document is loaded during the phone's startup process. Hence the application will be ready without delay when the user starts it. Please refer to the description of the **Auto start** parameter in Section 3.1.2, "Required Data".

The composition of the HTTP GET request sent by the phone depends on the target URL. If this URL has an `.xml` extension, the phone expects to receive a static XML document. In this case, the request URL will contain no additional information. Otherwise, if the requested URL has no `.xml` extension, the phone assumes it is communicating with a dynamic application, e. g. a servlet. If so, the phone will include its IP address, telephone number, and device type using key/value pairs in the request. The request URL has the following format:

```
http://<server ip address>:<port>/<program name>?ipaddress=
<phone IP>&phonenumber=<phone number>&devicetype=OpenStage
```

In both cases, whether the request URL has an `.xml` extension or not, the language currently selected on the phone is sent to the server in the `Accept-Language` field of the HTTP request header, for example:

```
Accept-Language: en
```

The following example shows a complete GET request sent by an OpenStage phone:

```
GET /ipp/example.txt?ipaddress=192.168.1.244&phonenumber=3338&device-
type=OpenStage HTTP/1.1
Host: 192.168.1.151\r\n
Connection: Close\r\n
Accept-Language: en\r\n
```

Application Suite Using Predefined Tabs

In addition to single applications, it is possible to run an application suite, which is divided in up to 3 internal tabs. With regular XML applications and phonebook applications, these tabs can be predefined by the administrator. For configuring an application suite, please refer to the descriptions of the tab-related parameters in Section 3.1, "Configuring an Application on the Phone".

The HTTP GET requests sent by the phone depend on the target URL. If this URL has an `.xml` extension, the phone will request the same XML document for each tab. If it has no `.xml` extension, the phone will append the unique tab name to the request URL. The request URI has the following format:

```
http://<server ip address>:<port>/<program name>?ipaddress=<phone
IP>&phonenumber=<phone number>&devicetype=OpenStage&tab=<unique tab
name>
```


Depending on how many tabs have been configured for the application, the phone will send 1 or 2 further similar requests for the other tabs. Each of these requests will be sent right after the response to the preceding request has been received. The focus will be on the last tab started. With Xpressions applications, the ☒ key is used for tab switching, and with other application types, the correspondent mode key or free programmable key is used.

Application Suite Using Tabs On Demand

As an alternative to predefined tabs, it is possible to start tabs on demand. The content for the tabs is delivered by the server using push requests (Section 2.3, "Push Mode" and Section 6.2, "The Push Request"). Xpressions always uses tabs on demand. With tabs on demand, it is possible to deliver different content for every tab. The tab that has been started first uses the start URL configured on the phone. The remaining tabs use the URL supplied by the remote server.

The focus is on the first tab started by the phone. With Xpressions applications, the ☒ key is used for tab switching, and with other application types, the correspondent key is used.

3.5 Mobile Users And Privacy

As some XML applications may process personal user data, it is important that the remote application is aware of the user currently logged onto the phone. This can be achieved by mapping the phone number against the phone's IP address. Both parameters are available to the remote program, as they are sent by the phone on application start. Please note that this will not be the case if the program name has an `.xml` suffix. For details please see Section 3.4, "Starting the Application".

3.6 User Interface

3.6.1 Display

When an XML application is started, an application tab appears underneath the status bar. It is labeled with the display name assigned to the application (see the **Display name** parameter in Section 3.1, "Configuring an Application on the Phone").

In the title bar, the title of the currently displayed XML document is shown. The title is an element within the structure of the currently displayed XML object.

The options bar provides a context menu for the commands associated with the XML objects in a screen. However, commands can be configured to be displayed only on the item they are associated with. If no command is configured to be displayed in the options bar, the options bar is not shown. For details about commands, please see Section 5.5, "Command".



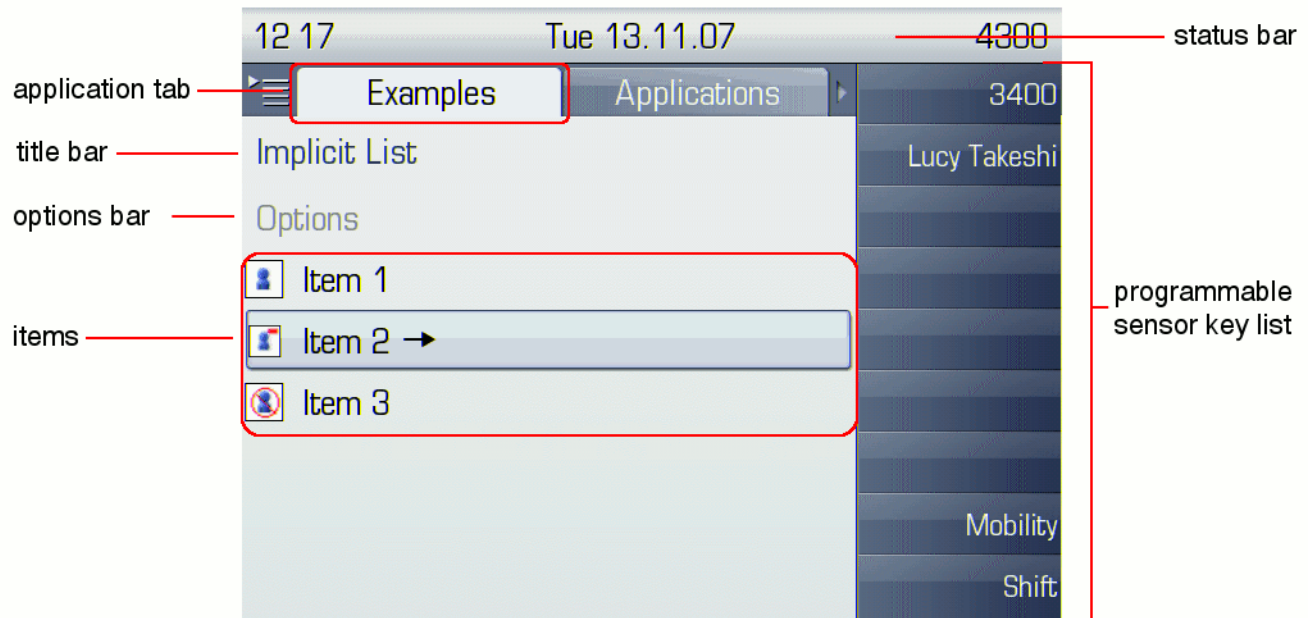
The label for the options bar depends on the current language setting for the phone. For details about setting the language, please refer to the OpenStage 60/80 User Guide.

Some of the XML objects described in Section 5, "XML Object Reference" are rendered as displayable items with which the phone user can interact, for instance item lists (see Section 5.8, "List"), text boxes (see Section 5.9, "Text Box"), or buttons (see Section 5.20, "Form/Button").

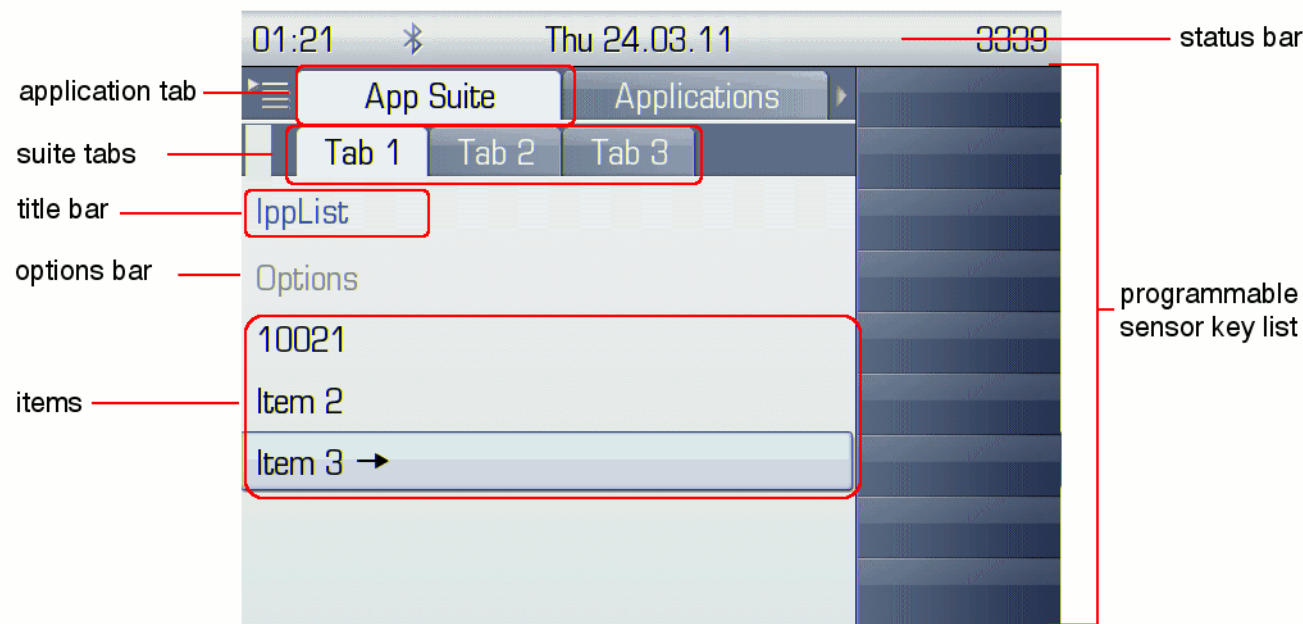


The screenshots in this document may deviate slightly from the current look of your phone display. This is due to possible skin or firmware changes.

Screenshot of a Single XML Application



Screenshot of an XML Application Suite



3.6.2 TouchKey

For navigating in an XML application, for selecting items, and to send commands and data to the server-side program, the TouchKey is used. The following table associates user operations and the corresponding function in the application.




Operation	Function
<p>▲ key or run finger counterclockwise</p> 	Navigate up, or to the previous item.
<p>▼ key or run finger clockwise</p> 	Navigate down, or to the next element.
<p>→ key</p> 	Open the right context menu

Table 3-1





Operation	Function
 key 	Depending on the context, one of these functions can be triggered: <ul style="list-style-type: none">• Open the left context menu• Leave the context menu• Go back to the previous screen
 key 	Execute a command in the context menu or directly associated with an item.

Table 3-1

3.6.3 Keypad

The keypad is used for entering alphanumerical data in input fields, or for direct keypad input in hotkey manner (see Section 5.6, "Direct Keypad Input"). For inputting alphanumerical data, the display keyboard is used; for use instructions, please refer to the OpenStage 60/80 User Guide or to the OpenStage Administration Manual.

3.6.4 TouchSlider (Sensor Slider)

In normal phone operation, the sensor slider is used for audio control. When an XML application is active, it can be used to set a numerical value. The value is read by means of the gauge object (see Section 5.21, "Form/Gauge").

3.6.5 LEDs

For some mode keys, the associated white LED of can be controlled by an XML application. Please refer to Section 5.4, "Action" and Section 3.1, "Configuring an Application on the Phone".

3.6.6 Additional Keys for Application Control

The free programmable function keys (FPKs) and, with firmware V2R1, the fixed function keys can be configured to send a customizable HTTP or HTTPS request to a URL (see Section 7, "Using Additional Phone Keys"). This provides an additional possibility to control the remote program that is associated with the XML application. The remote program can send feedback to the phone via a push request (see Section 6, "Push Capability").

4 XML Structure and Encoding



Please ensure that the XML documents used by the application do not exceed the maximum size of 25kB.

4.1 Functionality and Structure of an XML Application

The displayed items and input keys available for user interaction are coded as XML objects. In a valid XML document, these objects are arranged in container elements. The following table gives an overview of the individual objects and their functions:

Category	Object	Functional Description
Basic XML structure	Phone	Optional root element; reserved for future development.
	Display	Root element for objects within the display area.
Navigation	Command (BACK, SCREEN, UPDATE)	BACK: Leads to the screen displayed previously. SCREEN: Leads to a specific screen. UPDATE: Reloads the screen from the server. Displayed in context menus.
Form	Form	Container element that provides a form for sending the contents of data fields to the server.
Message window	Alert	Popup window which appears over the currently displayed screen and displays a message. Can be closed by user or timeout. Displayed over the other items on the main screen.
Ticker	Ticker	Ticker text moving from left to right. Displayed underneath the title bar.
Text, layout, image	Form/String Item	Shows text on the display. Displayed as an item on the main screen.
	Image	Displays an image file. Displayed as an item on the main screen.
	Form/Image Item	Positions an image.
	Form/Spacer	Creates a vertical space. Displayed as an empty line on the main screen.

Table 4-1

Category	Object	Functional Description
Selection lists	List	List of selectable options. Displayed as items on the main screen.
	Form/Choice Group	List of selectable options. Displayed as items on the main screen.
Data input	Text Box	Text input field with multiple lines. Displayed as singular item on the main screen.
	Form/Text Field	Text input field with a single line. Displayed as an item on the main screen.
	Form/Date Field	Date input field. Displayed as an item on the main screen.
	Form/Gauge (interactive)	Entry of a numerical value using a horizontal slider. Displayed as an item on the main screen.
Hidden data transfer	Hidden	Invisible field for transmitting preset data from the phone to the server.
Visualization of chronological processes	Form/Gauge (non-interactive)	Horizontal bar that grows every second. Displayed as an item on the main screen.
Sending data to the server	Command (SELECT)	The data contained in the entry fields on the screen are sent to the server. Displayed in context menus or associated with a list item.
	Form/Button	Sends data stored in its attributes, as well as data in a hidden object, to the server. Displayed as an item on the main screen.
Telephony	Action (MAKE-CALL)	Initiates a call to a specific telephone number.
LED control	Action (TURNLEDON or TURNLEDOFF)	Turns on/off the LED on the Message mode key

Table 4-1

4.2 Encoding/Internationalization

The OpenStage XML interface supports UTF-8 and ASCII. If a different character set is used, e. g. ISO 8859-1, special characters can be encoded by the appropriate numeric character references.

Example: `ä` will be rendered as the german umlaut "ä".

For details, see <http://www.w3.org/TR/2000/REC-xml-20001006#sec-references>.



Some text editors add the leading sequence 0xEF 0xBB 0xBF to an UTF-8 file. This is a Byte-Order Mark (BOM), which is conventionally used to indicate that the text is encoded as UTF-8. As the phone's XML parser does not support UTF-8 files with BOM, please make sure you use an editor that does not insert this mark. The problem exists, for instance, with older versions of Notepad++, and with Windows XML Notepad 2007. There are reportedly no problems with Notepad2, Notetab, or Altova XML Spy.



Please ensure that the encoding specification in the XML header is in capital letters, e. g. `<?xml version="1.0" encoding="UTF-8" ?>`

URLs sent by the phone are UTF-8 encoded; reserved characters are percent-encoded. The BNF grammar is used according to <http://www.ietf.org/rfc/rfc2396.txt>, appendix A.

5 XML Object Reference

This chapter describes the usage, content and error handling for the individual XML objects. The objects are comprised of a main element, its attributes and one or more child elements and their attributes. The names of the basic XML elements are prefixed with "Ipp", which stands for "IP phone"; for example, `IppAction`.

For each individual XML object, an example is provided. All examples from this document are available for download from the Siemens Enterprise Communications Community Portal. For downloading,

1. register with the Community Portal under
<http://www.siemens-enterprise.com/developerportal/Developer-Community/Register.aspx>
2. and navigate to
<https://app-enterprise.siemens-enterprise.com/gdmsproxy/retrieve?id=40777895>

5.1 Phone

`IppPhone` is an optional root element which enables the integration of special objects in the XML application. These objects are not displayed. They will be available in future versions of the OpenStage 60/80 phone software. Using `IppPhone` in current XML applications has the advantage of allowing easy extension in the future.

The use of `IppPhone` as the root element is not mandatory, `IppDisplay` (see Section 5.2, "Display") can also serve as the root element of an XML application.

Syntax

```
<IppPhone>  
  <IppDisplay/>  
</IppPhone>
```


5.2 Display

This object enables multiple screens within a single XML document. A "screen", in the context of this guide, means the content that is simultaneously shown on the display, comparable to a web page. The `IppDisplay` element functions as the root element of an XML document (unless `IppPhone` is used; see Section 5.1, "Phone"). Therefore, as the container for screen objects, the `IppDisplay` element is comparable to a deck of cards in a WML document.

An XML document may contain up to 5 screens, one of which may be the `InitialScreen` and another the `UpdateScreen`. If there are more than 5 screens in the XML document, the surplus screens are ignored.

Attributes

The following attributes are available to the `IppDisplay` element:

Attribute	Value	Functionality/Remarks
<code>InitialScreen</code> (optional)	ID of the initial screen (integer)	Used when there are multiple Screens within the same XML document. Determines which Screen is shown when the XML document is opened. If the attribute is not present, the initial Screen displayed is the first Screen in the XML document.
	0 or greater than the number of Screens, or negative, or not a valid number	The first Screen in XML document order is displayed as initial Screen.
<code>InitialScreen</code> (optional)	ID of the first screen to be displayed	The screen whose ID is given here is displayed when the document is loaded.
	Missing, negative, or invalid	The first screen in document order is displayed when the document is loaded.
<code>UpdateScreen</code> (optional)	ID of the second screen to be updated (integer)	Used when two screens have to be updated simultaneously. Provides the ID of the second screen to be updated.
	Missing, negative, or invalid	There is no second screen to update.

Syntax

```
<IppPhone>
  <IppDisplay UpdateScreen="[ID]">
    <IppScreen/>
    <IppScreen/>
    <IppScreen/>
    <IppScreen/>
    <IppScreen/>
  </IppDisplay>
</IppPhone>
```


5.3 Screen

The `IppScreen` element serves as a container for displayable items and commands. Switching between different screens contained in a document, i. e. an `IppDisplay` element, is possible using the `ID` attribute. The screen switching is done via the `SCREEN` command (see Section 5.5, "Command").



If a new screen is loaded whose screen ID already exists, elements of cached screens may be overwritten. For details, see the table in Section 2.4, "Cache".

Attributes

The following attributes are available to the `IppScreen` element:

Attribute	Value	Functionality/Remarks
<code>ID</code> (optional)	ID of the screen element (integer)	Enables referencing of the screen object, so that the screen can be accessed directly. If missing, negative, or invalid, it will default to 0.
<code>HiddenCount</code> (mandatory)	Number of Hidden objects	All hidden objects (see Section 5.11, "Hidden") are displayed.
	0	All hidden objects are ignored. If missing, negative, or invalid, it will default to 0.
	A number lesser than the actual number of Hidden objects	All the hidden objects that exceed the number indicated here are ignored.
<code>CommandCount</code> (mandatory)	Number of Command objects	All command objects (see Section 5.5, "Command") are displayed. If missing, negative, or invalid, it will default to 0.
	0	All command elements are ignored.
	A number lesser than the actual number of Command objects	All the command elements that exceed the number indicated are ignored.

Syntax



The IppScreen element must contain one of the following elements:

- IppAlert
- IppList
- IppTextBox
- IppForm

```
<IppPhone>
  <IppScreen ID="[ID]" HiddenCount="[Number]" CommandCount=
"[Number]">
    <IppKey/>
    <IppAction/>
    <IppCommand/>
    <IppAlert/>
    <IppList/>
    <IppTextBox/>
    <IppTicker/>
    <IppHidden/>
    <IppForm/>
    <IppPlayer/>
  </IppScreen>
</IppPhone>
```


5.4 Action







Actions provide a request mechanism to the phone to perform specified tasks: make a call to a specified number and end the call, or control mode key LEDs.



For security reasons, an XML application can only clear a call that it has initiated. Thus, in order to be able to clear a call down, the application must be running whilst the call is in progress. If the application is stopped and restarted whilst the call is in progress, the newly started application can not clear the call, because this new instance of the application did not initiate the call.

Attributes

For the `IppAction` element, the `TYPE` attribute is required with the following values:

Attribute	Value	Functionality/Remarks
TYPE (mandatory)	MAKECALL	A call to the telephone number specified in the child element <code>Number</code> is initiated. The possible formats for the call number depend on the phone's dial settings. Please refer to the OpenStage Administration Manual.
	ENDCALL	Ends the call made by a <code>MAKECALL</code> action.
	TURNLEDON	Turns on the white LED of the mode key on which the application is configured. (The white LED is always switched off when the blue LED is on, so that the white LED will only be visible when this application does not have the focus.) With firmware earlier than V2R1, this function is available only for Xpressions applications. Thus, only the white LED of the messages key  can be controlled. With firmware from V2R1 onwards, the white LED of the messages key  can be controlled with an Xpressions or messages application, and the call log key  can be controlled with a call log application.
	TURNLEDOFF	Turns off the white LED of the mode key on which the application is configured. (The white LED is always switched off when the blue LED is on.) With firmware earlier than V2R1, this function is available only for Xpressions applications. Thus, only the LED of the messages key  can be controlled. With firmware from V2R1 onwards, the white LED of the messages key  can be controlled with an Xpressions or messages application, and the call log key  can be controlled with a call log application.
	Missing or invalid	No action will be performed.

Syntax

```
<IppAction Type="MAKECALL | ENDCALL | TURNLEDON | TURNLEDOFF" >
  <Number>[Phone Number]</Number>
</IppAction>
```


XML Object Reference

Action

Child elements

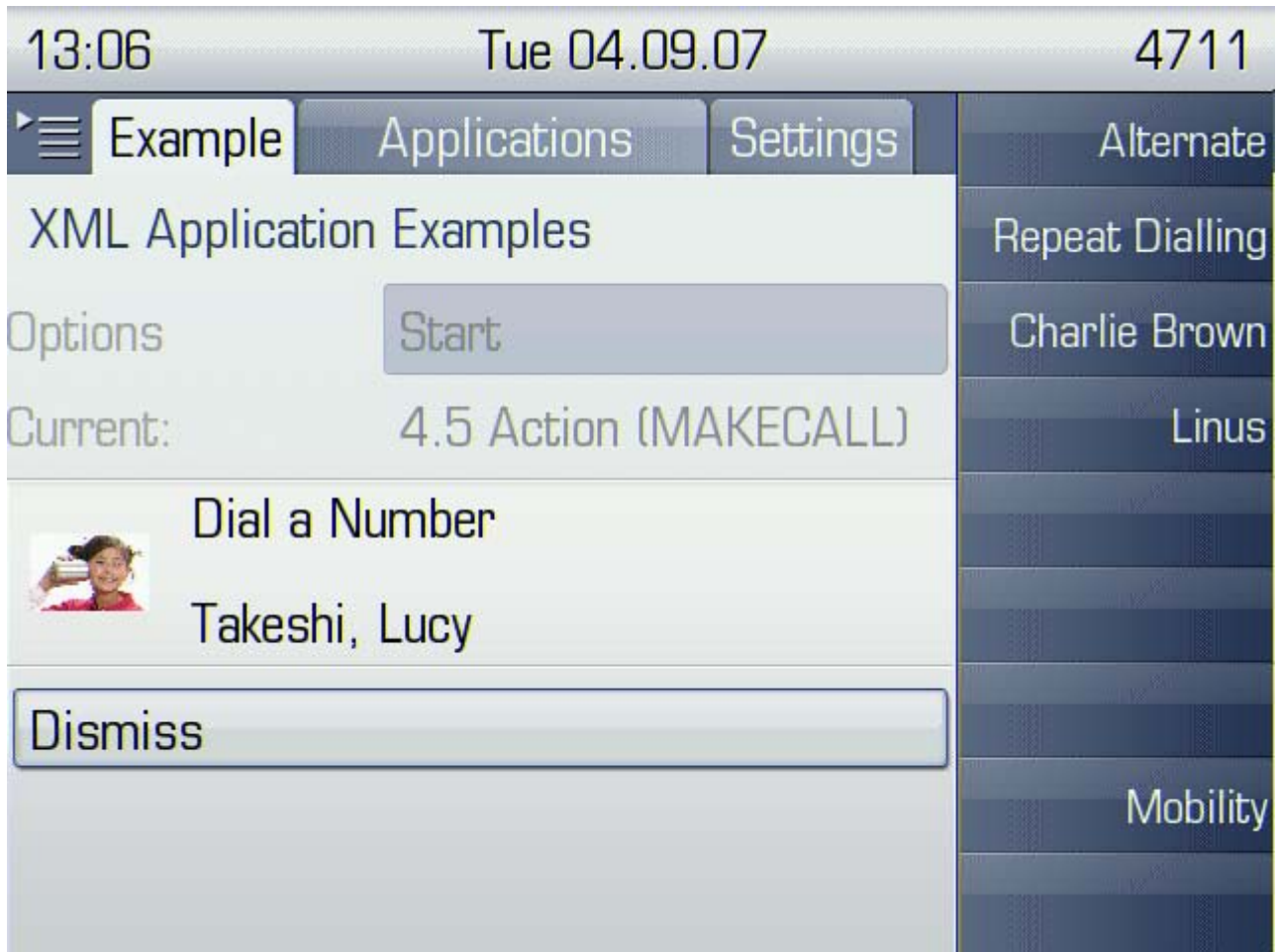
- Number

The telephone number to be dialed.




Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="0">
      <IppAlert Type="CONFIRMATION" Delay="FOREVER">
        <Title>Dial a Number</Title>
        <Text>
          3290
          <IppPhoneNumber ImageType="PICTURECLIP"
NumberType="NAME">
            <AltText>No image available for this entry</AltText>
          </IppPhoneNumber>
        </Text>
        <Image></Image>
      </IppAlert>
      <IppAction Type="MAKECALL">
        <Number>+49897223290</Number>
      </IppAction>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```



Screenshot



5.5 Command

Commands generally appear in context menus which are associated either with the options bar or with the clickable area of a specific item. There are 2 different context menus: a click on the  key invokes the left context menu, and a click on the  key invokes the right context menu. The user can leave the context menu by pressing the  key.

The right context menu shows all commands, according to the settings of the `DisplayOn` attribute. The left context menu only shows CANCEL, BACK, and EXIT commands. For more information, please refer to the attribute table underneath.

If a command is specified accordingly and associated with an implicit list, it can be started directly with the  key (see the attribute table underneath, and Section 5.8, "List").

With alerts (see Section 5.7, "Alert"), the commands are displayed at the bottom of the alert.



Attributes

The following attributes are available to the `IppCommand` element:

Attribute	Value	Functionality/Remarks
Type (mandatory)	SELECT	Transmits the key/value pairs from the current list, text box, form, or player, to the URL indicated in the relevant object. Key/value pairs from hidden objects and the key/value pair associated with the command, if present, are also sent to the server.
	BACK	Switches to the cached version of the screen displayed previously. The command will always go back to the previous full screen, i. e. form, list, or text box. If the previous screen is an alert (see Section 5.7, "Alert") or player (see Section 5.23, "Player"), it will be skipped. If there is only a single screen in the XML document, no menu item is generated for this command.
	UPDATE	Replaces the current screen with the screen specified in the current list, text box, form, or player. The data for this screen is fetched from the server using the URL associated with the current screen. Key/value pairs from hidden elements and the key/value pair associated with the command, if present, are sent to the server. As opposed to a SELECT Command, no key/value pairs from the current list, text box, form, or player, are sent to the server.
	SCREEN	Switches to the screen specified in the child element <code>ScreenID</code> . This screen must be contained in the same display element (<code>IppDisplay</code>). The server-side program is not informed of this action. If the data in <code>ScreenID</code> are missing or invalid, the current screen will continue to be displayed.
	CANCEL	Used to undo a selection made in selection lists (see Section 5.8, "List", and , Form/Choice Group) or an entry made in input fields (see Section 5.9, "Text Box", Section 5.17, "Form/Text Field", and Section 5.19, "Form/Date Field"). The contents of the current screen revert back to their preset values cached on the phone.
	EXIT	Ends the application.

XML Object Reference

Command

Priority (optional)	Priority (integer)	Determines the order of the command objects, whereby 0 represents the one with the highest priority, followed by 1 and so on.
Auto (optional)	Total duration in seconds	The command object is automatically executed after the specified time, that is, without the user doing anything. If multiple command objects containing this attribute exist, the one that is set for the shortest total duration is executed. This means that only one command per screen is executed automatically.
	0 or missing, or negative, or invalid	The command is only invoked through user action.
Key (optional)	Key	The key for the key/value pair that is sent to the server-side program. Please note that the key must be unique within a screen.
Value (optional)	Value	The value for the key/value pair that is sent to the server-side program.
DisplayOn (optional)	OPTIONS	Default value. The command is displayed from the options bar.
	LISTITEM	The command is displayed on a list item (see Section 5.8, "List") or on an individual item in a form, if included in a form item (see Section 5.13, "Form/Item").
	BOTH	The command is displayed both on the options bar and on list items or form items.
Select (optional)	YES	The command is pre-selected in an implicit list (see Section 5.8, "List"). If more commands are pre-selected, only the first one in document order will be used. Pressing  when the options bar is highlighted returns the key/value pair for the pre-selected command. Pressing  when an individual list item is highlighted returns the key/value pairs for both the pre-selected command and the highlighted list item.
	NO	Default value. The command is not pre-selected in an implicit list.

Default (optional)	YES	The command is set as the default command in the context menu. If more than one command is set as the default command, only the first one will be the default command.
	NO	Default value. The command is not set as the default command in the context menu.

Syntax

```
<IppCommand Type="SELECT|BACK|UPDATE|SCREEN|CANCEL|EXIT"
Priority="[Number]" Auto="[Number]" Key="[Key]" Value="[Value]"
DisplayOn="OPTIONS|LISTITEM|BOTH" Select="YES|NO" Default="YES|NO">
  <Label> </Label>
  <ScreenID> </ScreenID>
</IppCommand>
```



Multiple occurrences of a command type may appear on a screen. In such cases, the commands should have different labels in order to be distinguishable by the user.



If the `Type` attribute is missing or has an invalid value, the command is not displayed. If the child element `Label` is missing, the command type is displayed instead of a label.

Child elements

- Label

Contains the label for the menu item that is generated for the Command.

- ScreenID

Indicates which screen should be accessed next on activating a `SCREEN` Command. The `ScreenID` is only appropriate if the Command type is `SCREEN`. If the `ScreenID` is missing or invalid, the current Screen will not be replaced, but continue to be displayed.

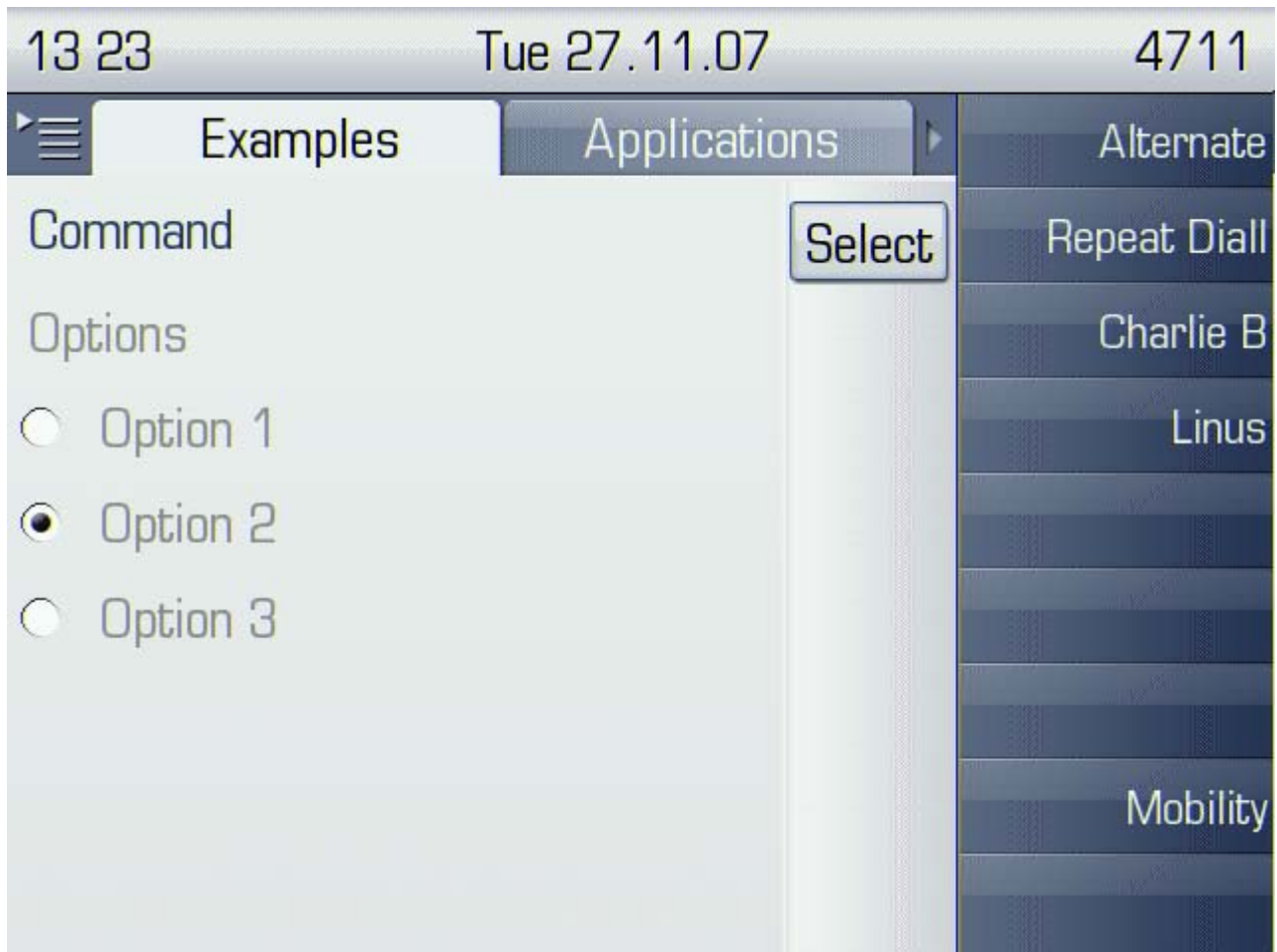
XML Object Reference

Command

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="2">
      <IppList Type="EXCLUSIVE" Count="3">
        <Title>Command</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <Option ID="1" Selected="FALSE" Key="key" Value="1">
          <OptionText>Option 1</OptionText>
        </Option>
        <Option ID="2" Selected="TRUE" Key="key" Value="2">
          <OptionText>Option 2</OptionText>
        </Option>
        <Option ID="3" Selected="FALSE" Key="key" Value="3">
          <OptionText>Option 3</OptionText>
        </Option>
      </IppList>
      <IppCommand Type="SELECT" DisplayOn="BOTH" Priority="1">
        <Label>Select</Label>
        <ScreenID>2</ScreenID>
      </IppCommand>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```


Screenshot



5.6 Direct Keypad Input

5.6.1 Overview

In addition to regular user input by the means of input fields, it is possible to send keypad entries to the remote server in realtime, or, alternatively, in buffered mode. In unbuffered mode, the phone will immediately send an HTTP GET request to the server as soon as a keypad key has been pressed. After this, the phone will wait for a server response. The request contains a key/value pair wherein the key is predefined in the XML document, and the value represents the value of the keypad key.

The phone itself will not generate audible or visual feedback to the user when the data is sent; this must be handled by an appropriate server response.

The feature is available exclusively for lists (see Section 5.8, "List"); with alerts, forms, text boxes, or players, only regular user input is supported. To enable and configure direct key input, the `IppKey` element is used.

To add context information, hidden (Section 5.11, "Hidden") objects can be used. The key/value pairs contained in the hidden objects will be sent along with each keypad input. For example, the remote server may keep track of the entire digit sequence entered so far by including this digit sequence in an hidden object.



Long presses of the * and # keypad keys toggle the ringing state and invoke phone lock. As the keypad input feature does not distinguish between short and long press, these functions would occur in parallel with sending keypad input to the remote server. If the remote server updates an XML application whilst the phone is locked, the user does not see the updated screen until after the phone is unlocked.



Certain keypad key combinations may be invoked on the phone to enter the admin menu, invoke a factory reset, or restart the phone. When the first key of a combination is pressed, the corresponding value will be sent to the remote server. However, the subsequent key presses of the key combination will not be communicated.



If an alert (see Section 5.7, "Alert") is displayed over a list which had sending keypad input enabled, this function will be automatically disabled. Moreover, it will stay disabled when the alert is dismissed and the list has focus. To re-enable sending keypad input on the list, the XML document for the list must be sent again.

Buffering

As an alternative to immediate sending and interaction with the server, keypad input can be buffered and sent afterwards. This is controlled by the `BufferKeys` element. Depending on the settings of `BufferLength` and `TermKey`, the data is submitted either when the key sequence has reached the predefined buffer length, or when the predefined termination key is pressed.

If an application requires each keypad key press one at a time, the remote server should request subsequent buffering (see Section 5.6.3.4, "Buffer Subsequent Keypad Key Presses"). This ensures that no keypad key presses are thrown away when the user types keypad keys quicker than responses are received from the server.

Error Handling

If there is a failure in the communication protocol between the phone and the remote server when sending keypad key presses to the remote server, no error message is displayed to the user. However, if requested in the XML document, the key presses will be buffered on the phone. When the user presses the next keypad key, and the connection is re-established, that key press and all buffered key presses will be sent to the remote server.

5.6.2 Possible Uses

Potential example XML applications using sending keypad input are:

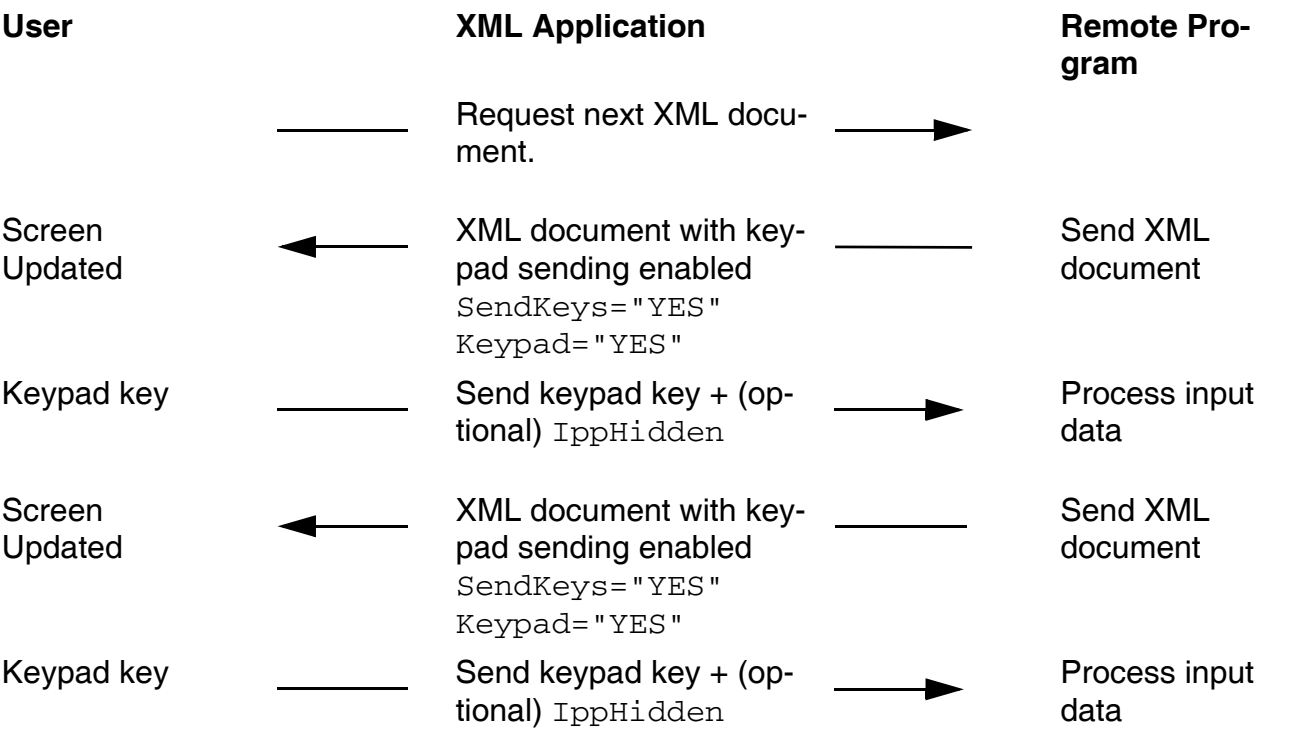
1. A phonebook application in which the user initiates a search for the appropriate entry using the keypad keys. The digits are sent to the remote program. The remote program uses the digits as search criteria and returns a list of matching entries. These entries are immediately displayed to the user. If the user presses subsequent keypad keys after the first keypad digit, and a response has not yet been received, the keypad digits are buffered, and sent en bloc to the remote program.
2. A menu application provides a sequence of numbered lists of options. The user navigates in a list using the keypad keys. The remote program requires single keypad digits to determine the next list. Therefore, multiple keypad digits or any subsequent keypad digits are ignored whilst the phone is waiting for server response.
3. An application requests the user to enter a PIN. The user presses a sequence of keypad digits terminated by #. The digits are buffered until the # is detected. After this, all of the digits including the # are sent en bloc to the remote program.

5.6.3 Typical Client-Server Interactions

This section describes typical interactions between the phone-side XML application and the server-side program with different settings for the keypad sending function.

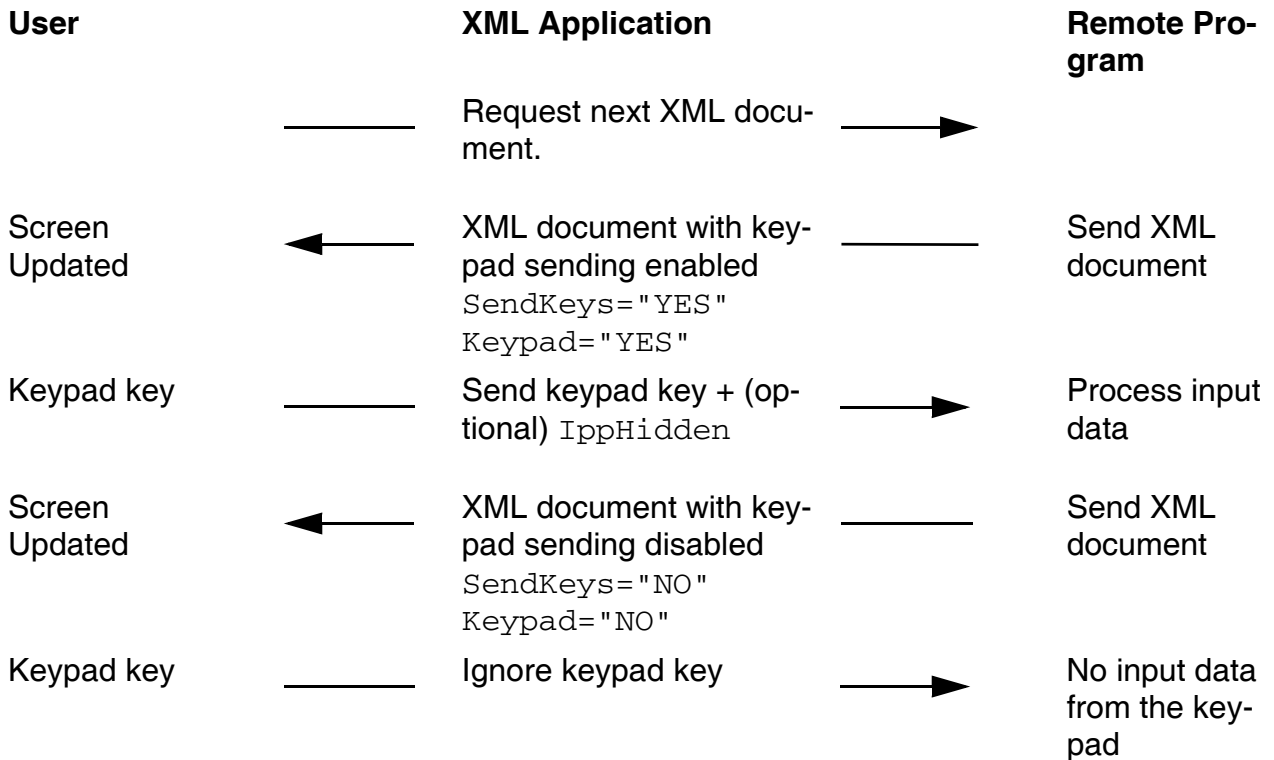
5.6.3.1 Sending Keypad Keys Enabled and Immediate Submission

In the following interaction, the remote program enables sending keypad input and disables buffering. As soon as the user presses a keypad key, the screen updates.



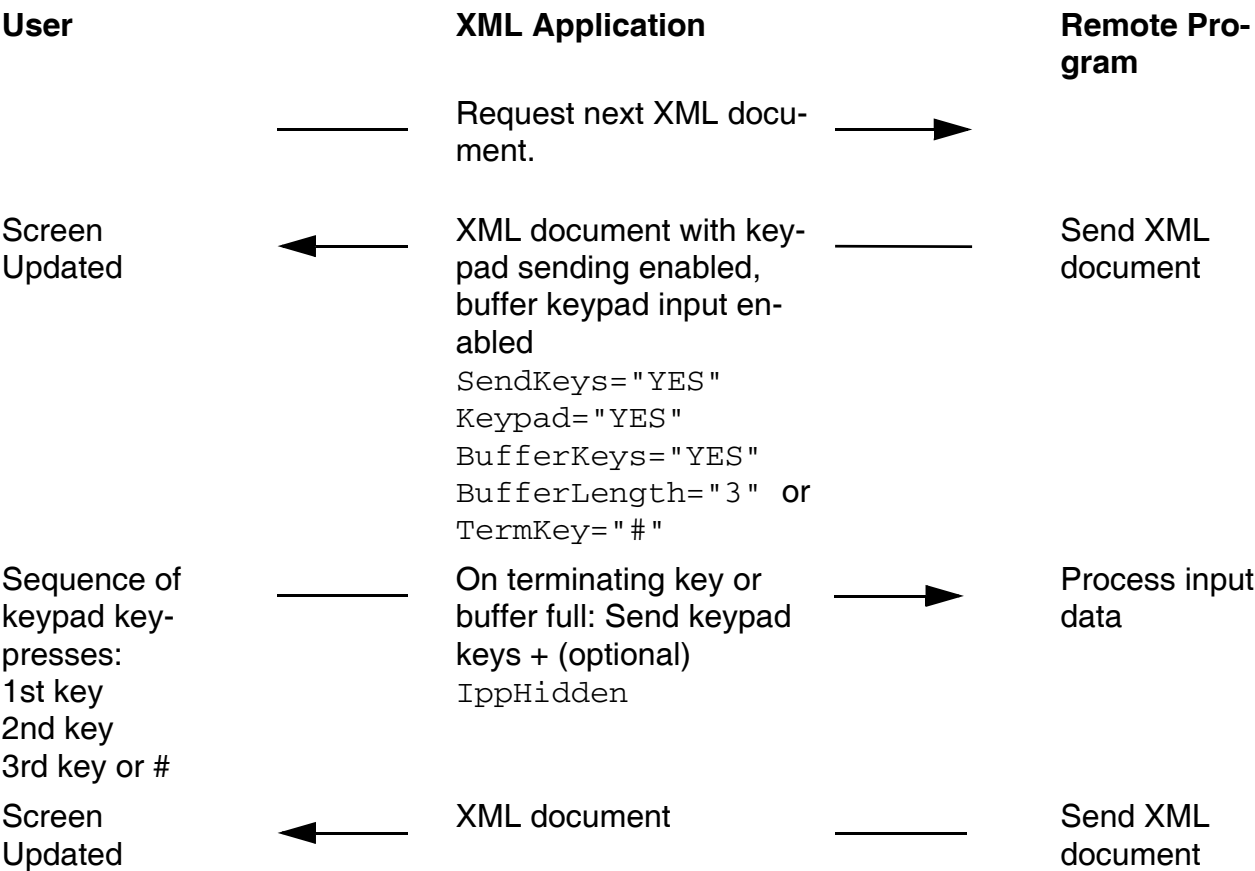
5.6.3.2 Disable Sending Keypad Keys

In the following interaction, the remote XML application initially enables sending keypad input. The user presses a keypad key which is sent to the remote program. With the following XML document, the XML application disables sends keypad input. All subsequent keypad key presses are ignored.



5.6.3.3 Buffer Initial Keypad Key Presses

In the following interaction, the remote program enables send keypad input and buffering with a buffer length greater than 1. When the user presses one or more keypad keys, the key presses are buffered until the buffer is full, i.e. the number of key presses equals the predefined buffer length. After this, the key data are sent en bloc to the remote XML application. Alternatively, a terminating keypad key is defined and the non-terminating keypad key presses are buffered until the terminating key is pressed. After this, the key data are sent en bloc, including the terminating keypad key.

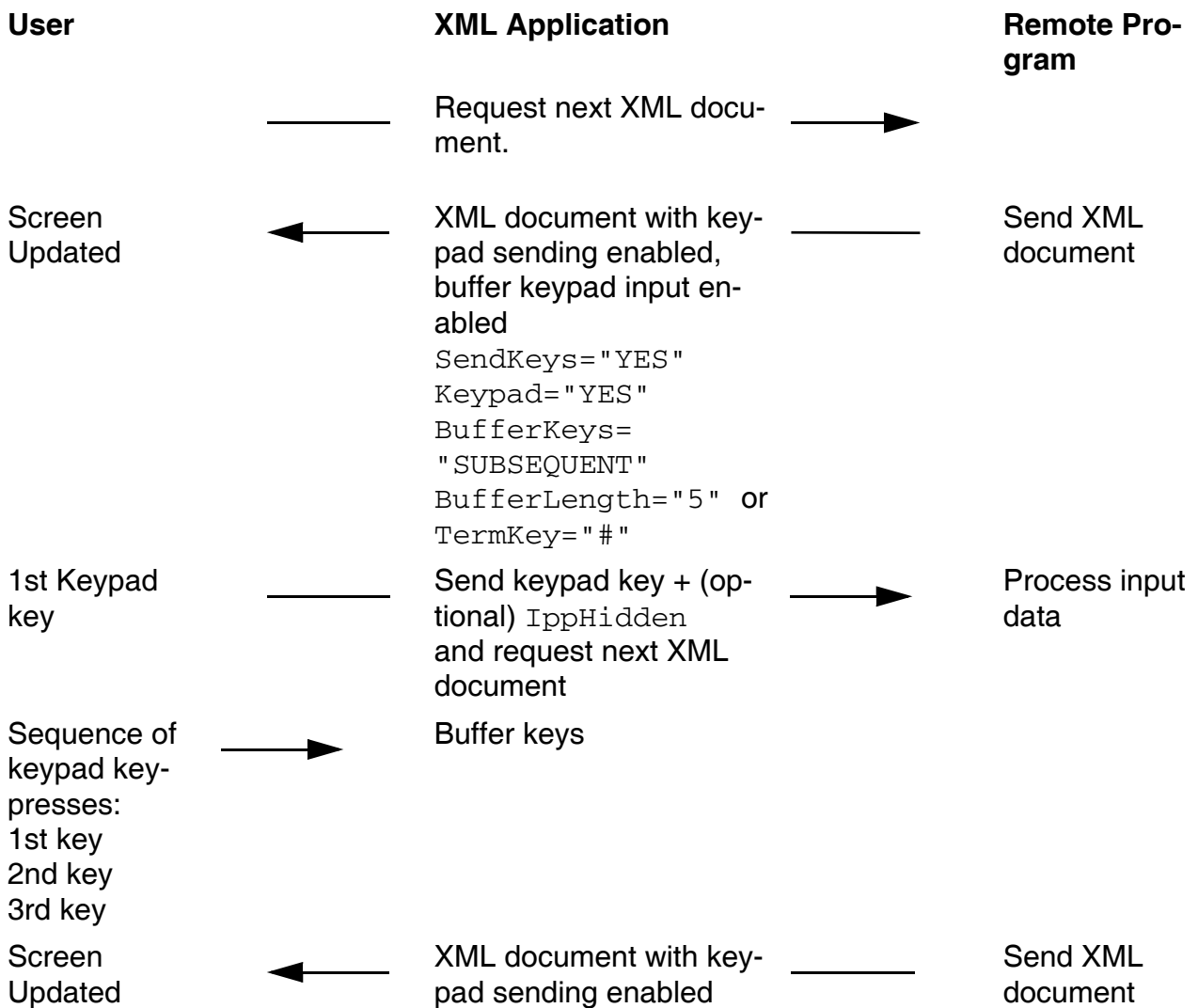


5.6.3.4 Buffer Subsequent Keypad Key Presses

In the following interaction, the remote program enables sending keypad input and buffering subsequent keypad keys. As soon as the user presses the first keypad key, e. g. to enter the first letter of a name to be searched in a phonebook, the key is sent to the server. Subsequently, the user types in some more letters or digits. These characters will be buffered until:

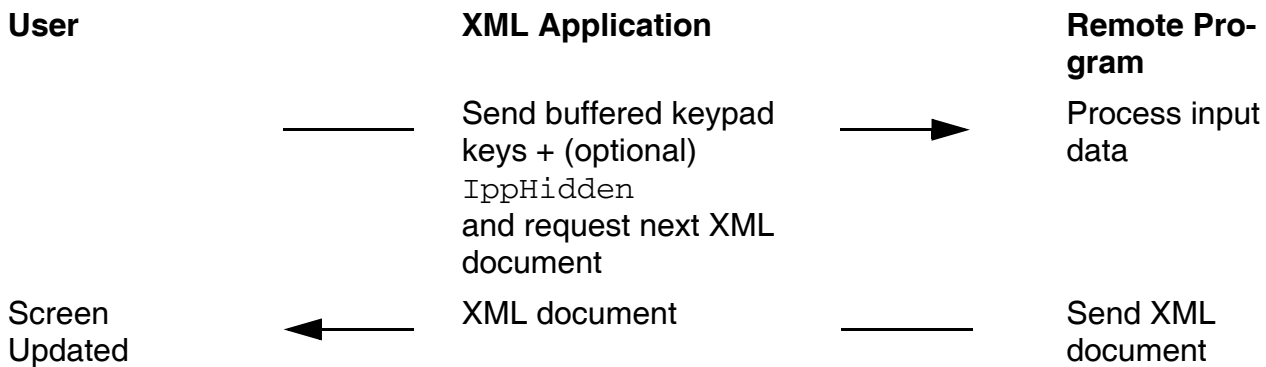
- the buffer length is reached, or
- the terminating key, if defined, has been pressed.

As soon as one of these criteria is met, and the response to the previous request issued with sending the first key has been received, the buffered keys are sent to the server. In the example interaction, the response from the server is received first, that is, before the buffer length is reached or the user has typed in the terminating key.



XML Object Reference

Direct Keypad Input



Attributes

The following attributes are available to the `IppKey` element:

Attribute	Value	Functionality/Remarks
Keypad (mandatory)	YES	Keypad key presses are handled.
	NO	Default value. Keypad key presses are ignored.
SendKeys (mandatory)	YES	Key digits are sent.
	NO	Default value. Key digits are not sent.
BufferKeys (optional)	YES	The key presses are buffered.
	NO	Default value. The key presses are not buffered, but immediately sent to the server.
	SUBSEQUENT	The key presses following the first key press are buffered.
BufferLength (optional)	Number of key presses to be buffered Default: 0	Defines the length of the buffer. When the buffer is full, the digits are sent en bloc.
TermKey (optional)	0 to 9, *, #, or left blank Default: blank	Defines the terminating keypad key, which triggers the submission of the buffered digits. Please note that the character # is URL encoded.
UrlKey (mandatory)	User defined string Default: digit	Defines the key part of the key-value pair sent in the URL string to the remote server.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
  <IppDisplay InitialScreen="1">
    <IppScreen ID="1" HiddenCount="1" CommandCount="1">
      <IppKey Keypad="YES" SendKeys="YES" BufferKeys="YES" Buffer-
Length="3" UrlKey="mydigit"/>
      <IppList Type="IMPLICIT" Count="3">
        <Title>IppList</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <Option ID="1" Selected="FALSE" Key="1" Value="1">
          <OptionText>Item 1</OptionText>
          <Image/>
        </Option>
        <Option ID="2" Selected="TRUE" Key="2" Value="2">
          <OptionText>Item 2</OptionText>
          <Image/>
        </Option>
        <Option ID="3" Selected="FALSE" Key="3" Value="3">
          <OptionText>Item 3</OptionText>
          <Image/>
        </Option>
      </IppList>
      <IppCommand Type="SELECT" Priority="0">
        <Label>Main Screen</Label>
        <ScreenID>2</ScreenID>
      </IppCommand>
      <IppHidden Type="VALUE" Key="xmlobject">
        <Value>_show</Value>
      </IppHidden>
    </IppScreen>
  </IppDisplay>
```


5.7 Alert

An Alert is a message window displayed on the bottom half of the current screen. Different types of alert can be defined using the `Type` attribute. Title, text, and an image for the alert can be specified in the child elements.






An alert implicates a special command to close the alert. This command has the label "Dismiss". If the screen containing the alert also includes a command, the "Dismiss" command is replaced by the command explicitly added to the screen. In this case the alert can only be cleared by the user.

The optional element `Url` enables explicitly added `SELECT`, `UPDATE`, and `EXIT` commands (see Section 5.5, "Command") to be returned to the server-side program. If the explicitly added commands are only `BACK`, `SCREEN`, or `CANCEL`, the `Url` element is not required .

Another optional element, `IppPhoneNumber`, provides a connection to the local phone book. If the `IppPhoneNumber` element contains a phone number that exists in the phone's local phone book, the name and number of the participant are displayed. Furthermore, if available, a picture clip or, else, an icon for the phone type in question (e.g. mobile number, home number, etc.) are shown. A picture clip from the phonebook will override an image object. The translation of phone number to name, number, and picture/icon is done on the phone. If the translation fails, alternative text may be displayed instead of the phone number. The alternative text is supplied in the XML document. The `IppPhoneNumber` element is defined in Section 5.24, "Phone Number".

Attributes

The following attributes are available to the `IppAlert` element:

Attribute	Value	Functionality/Remarks
Type (optional)	ALARM	 An alarm icon is displayed at the left side.
	CONFIRMATION	 A prompt for confirmation icon is displayed at the left side.
	ERROR	 Default value. An error icon is displayed at the left side.
	INFO	 An info icon is displayed at the left side.
	WARNING	 A warning icon is displayed at the left side.
Delay (optional)	Timeout in milliseconds	After the specified time elapses, the alert disappears and the previously displayed screen reappears
	FOREVER	Default value. The display of the alert can be deactivated only by the user.
	Missing, negative, or invalid	Will default to <code>FOREVER</code> .

Syntax

```
<IppAlert Type="ALARM|CONFIRMATION|ERROR|INFO|WARNING"
Delay=" [Number] |FOREVER">
  <Title> </Title>
  <Url> </Url>
  <Text>
    <IppPhoneNumber> </IppPhoneNumber>
  </Text>
  <Image Cache="n"> </Image>
</IppAlert>
```

Child elements

- Title

The title of the alert is displayed in the title bar.

- Url

This URL is used by a `SELECT`, `UPDATE`, or `EXIT` command, if such a command is added to the alert.

- Text

Text in an alert. An `IppPhoneNumber` (see Section 5.24, "Phone Number") can be contained, which enables the automatic translation of a phone number stored in the local phone book into an icon, a picture clip or the name associated to the number.

- Image

If an image is provided, it will replace the standard symbol for the alert. For more information about the `Image` tag, see Section 5.22, "Image". Please note that, if the `Text` element contains an `IppPhoneNumber`, the picture clip or phone type icon from the phonebook will override the image object.

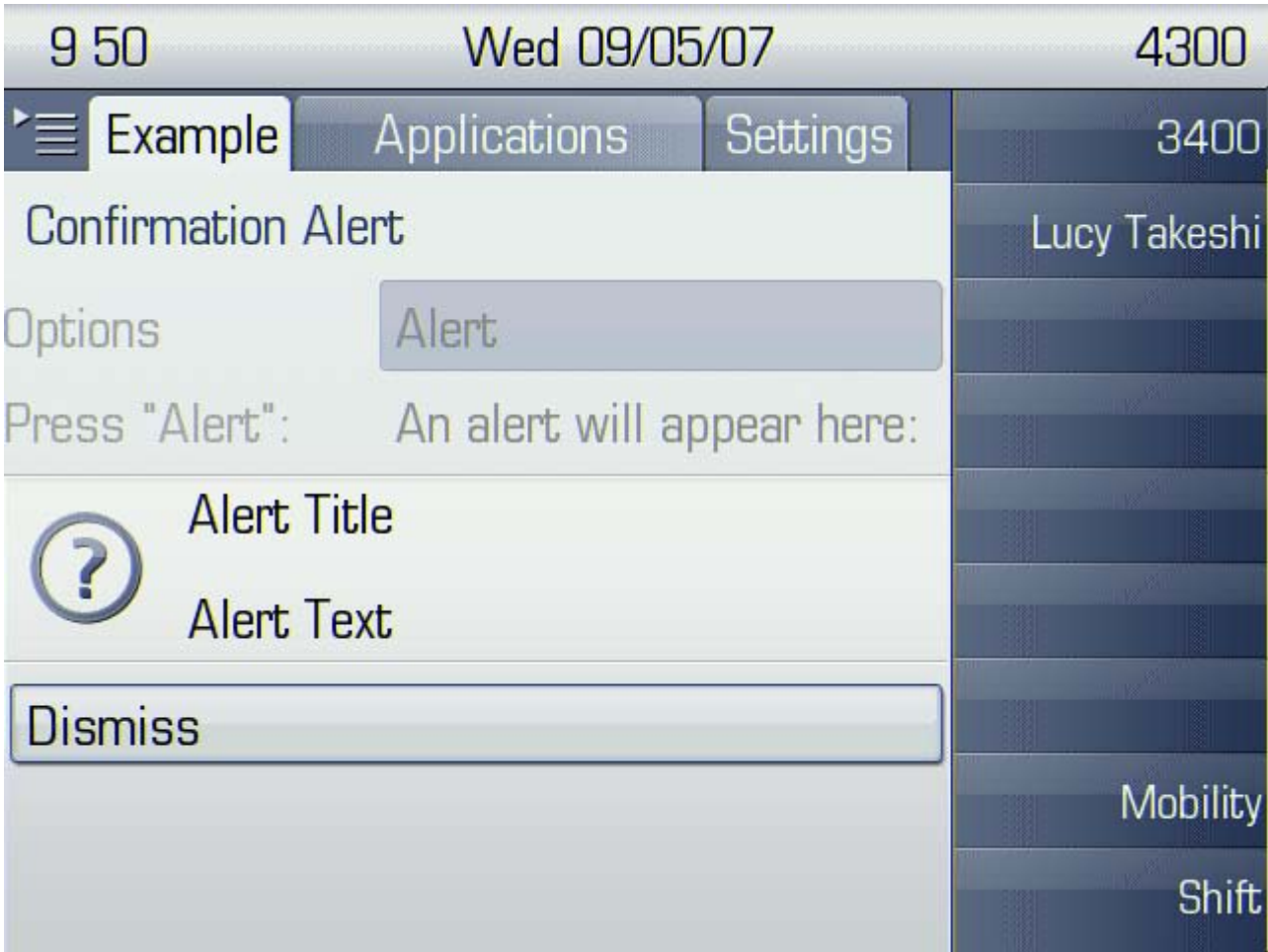
Commands

If one or more commands are added to an alert, i. e. to the screen containing the alert, these commands will replace the implicit "Dismiss" command.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="1">
      <IppForm ItemCount="1">
        <Title>Confirmation Alert</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <IppStringItem>
          <Label>Press "Alert":</Label>
          <Text>An alert will appear here:</Text>
        </IppStringItem>
      </IppForm>
      <IppCommand Type="SCREEN" Priority="0">
        <Label>Alert</Label>
        <ScreenID>2</ScreenID>
      </IppCommand>
    </IppScreen>
    <IppScreen ID="2" HiddenCount="0" CommandCount="0">
      <IppAlert Type="CONFIRMATION" Delay="FOREVER">
        <Title>Alert Title</Title>
        <Text>Alert Text</Text>
      </IppAlert>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```


Screenshot



5.8 List

This object represents a selection list, comparable to the `<select>` and `<option>` elements in HTML or WML. A screen can only contain one list object.

There are three types of selection lists: `IMPLICIT`, `EXCLUSIVE`, and `MULTIPLE`. While the types `IMPLICIT` and `EXCLUSIVE` permit the selection of only one single option, the type `MULTIPLE` permits the selection of one or more options. Depending on the type of list, the list items are displayed as plain list (`IMPLICIT`), or with radio buttons (`EXCLUSIVE`), or checkboxes (`MULTIPLE`). Additionally, images can be added to the list items (see Screenshot (1 column)).

Attributes

The following attributes are available to the element:

Attribute	Value	Function/Remarks
Type (mandatory)	<code>IMPLICIT</code>	Default value. As soon as an option has been selected, the data are sent immediately to the URL specified. Thus, exactly one option can be selected, and only the key/value pair that has been selected is sent to the server.
	<code>EXCLUSIVE</code>	Exactly one option can be selected, in radio button fashion. Only the key/value pair selected is sent to the server.
	<code>MULTIPLE</code>	One or more options can be selected. All key/value pairs present in the list are sent to the server. For selected options, the value of the <code>value</code> attribute is then sent; for non-selected options, this value is prefixed with <code>not_</code> .
Count (mandatory)	Number of options	Only as many options as indicated here are displayed. If the actual number of options exceeds this amount, all further options are ignored.
	0	No options are displayed.

Columns (optional)	Number of columns (1 or 3)	<p>A list may contain a single column or three columns of option text. The default value is 1, which results in a single column list, just like a list without this attribute. A single column list occupies 100% of the item width in a form. In a 3 column list, the width is distributed as follows:</p> <ul style="list-style-type: none">• left column: 50%• middle column: 16%• right column: 34% <p>Each option container must have the same number of <code>OptionText</code> elements as the number of columns. The <code>OptionText</code> elements are displayed in order from the left to right column. If <code>Columns</code> is missing, negative, or invalid or there is a mismatch between the number of columns and <code>OptionText</code> elements, the list defaults to a single column list, using only the first <code>OptionText</code> element in each <code>Option</code> container.</p>
-----------------------	----------------------------	--

Syntax

```
<IppList Type="IMPLICIT|EXCLUSIVE|MULTIPLE" Count="[Number]" Columns
="[Number]">
  <Title> </Title>
  <Url> </Url>
  <Option Selected="FALSE|TRUE" Key="[Key]" Value="[Value]">
    <OptionText State="GRAY|NORMAL">
      <IppPhoneNumber> </IppPhoneNumber>
    </OptionText>
    <Image Cache="[Name]"> </Image>
  </Option>
</IppList>
```


Syntax (3 columns)

```
<IppList Type="IMPLICIT|EXCLUSIVE|MULTIPLE" Count="[Number]" Columns
="3">
  <Title> </Title>
  <Url> </Url>
  <Option Selected="FALSE|TRUE" Key="[Key]" Value="[Value]">
    <OptionText State="GRAY|NORMAL">
      <IppPhoneNumber> </IppPhoneNumber>
    </OptionText>
    <OptionText State="GRAY|NORMAL">
      <IppPhoneNumber> </IppPhoneNumber>
    </OptionText>
    <OptionText State="GRAY|NORMAL">
      <IppPhoneNumber> </IppPhoneNumber>
    </OptionText>
    <Image Cache="[Name]"> </Image>
  </Option>
</IppList>
```

Child elements

- Title

Title text for the selection list. The title is displayed in the title bar.

- Url

The URL to which the data of the selected option are sent



If the URL is missing or is invalid, the current screen will continue to be displayed unchanged.


- Option


This contains the data that are sent to the server. It can also contain descriptive text (OptionText) and an image (Image).

The following attributes are available to the element :

Attribute	Value	Function
Selected (optional)	FALSE	Default value. The option is not preselected.
	TRUE	The option is preselected. When the IppList type is IMPLICIT, the option is highlighted.
Key (optional)	Key	The key for the key/value pair that is sent to the server as an HTTP POST request. For selection lists of type IMPLICIT and EXCLUSIVE, multiple options can have the same Key. If the type is MULTIPLE, all the values must be different.
Value (optional)	Value	The value for the key/value pair that is sent to the server as an HTTP POST request. If the selection list type is MULTIPLE, all the key/value pairs are sent. For selected options, the value of the value attribute is then sent; for non-selected options, this value is prefixed with not_.

Table 5-1

 If a key or value is missing or invalid, null is sent instead.

 The uniqueness of keys and values is not checked in the telephone.

- OptionText

Descriptive text for the option. An IppPhoneNumber (see Section 5.24, "Phone Number") can be included, which enables the automatic translation of a phone number stored in the local phone book into the name associated to the number, and an icon or a picture clip.

The following attribute is available to the element

Attribute	Value	Function
State (optional)	NORMAL	The text is displayed in regular color.
	GRAY	The text is displayed in gray color.

- Image

Illustrative image for the option. The image is displayed to the left of the text. For more information about the image element, see Section 5.22, "Image".

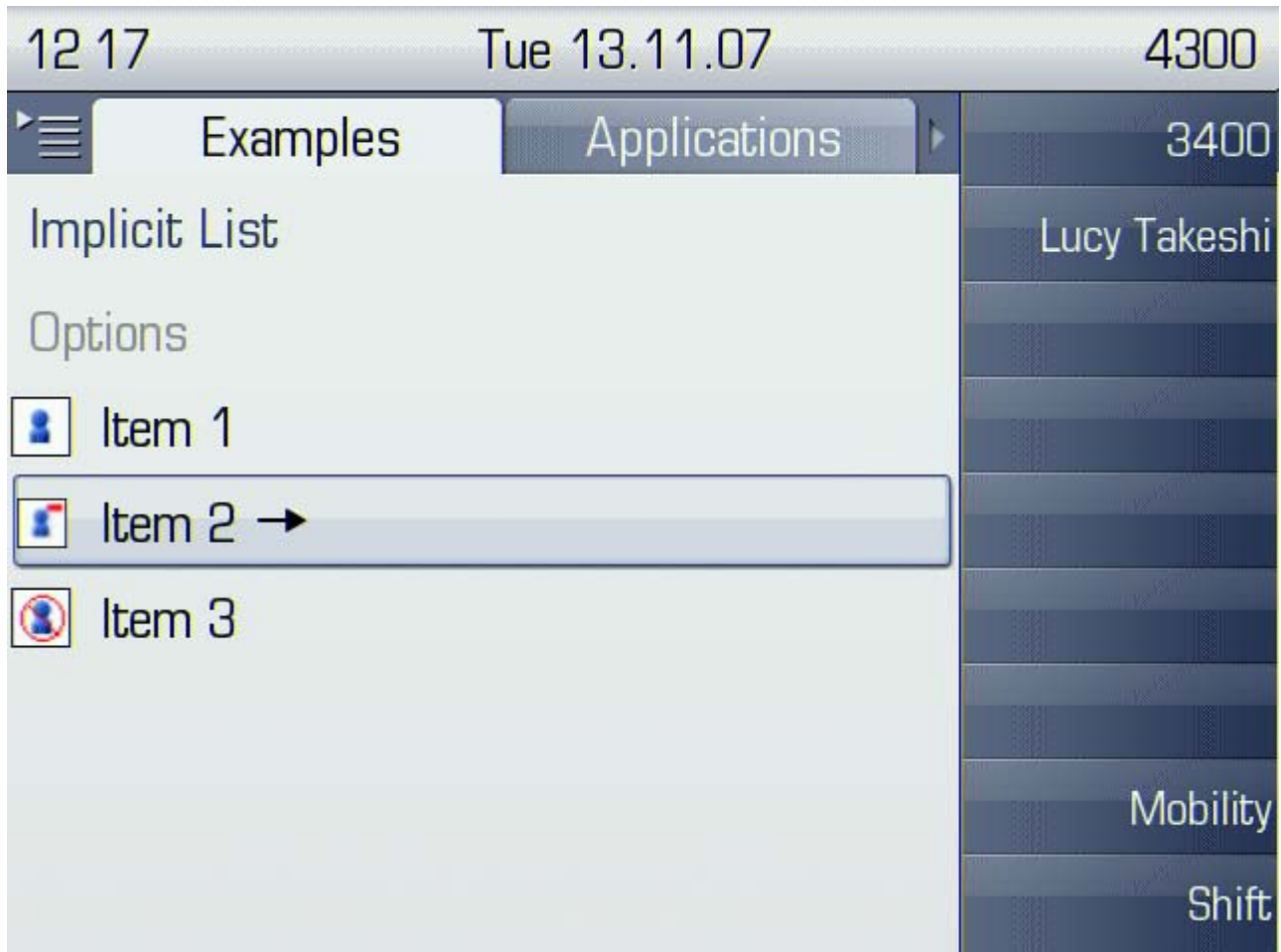
Commands

Unless the List type is `IMPLICIT`, a command is required to send data to a server and initiate server-side action. A command is added by inserting a command object in the screen (see Example (1 column)). The command can be displayed on the context menu of the options bar, or on the context menu of each individual item on the list, or both. For details, please see Section 5.5, "Command".

Example (1 column)

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="1">
      <IppList Type="IMPLICIT" Count="3">
        <Title>Implicit List</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <Option Selected="FALSE" Key="key1" Value="1">
          <OptionText>Item 1</OptionText>
          <Image>http://subdomain.domain/path/image1.png
        </Image>
        </Option>
        <Option Selected="TRUE" Key="key2" Value="2">
          <OptionText>Item 2</OptionText>
          <Image>http://subdomain.domain/path/image2.png
        </Image>
        </Option>
        <Option Selected="FALSE" Key="key3" Value="3">
          <OptionText>Item 3</OptionText>
          <Image>http://subdomain.domain/path/image3.png
        </Image>
        </Option>
      </IppList>
      <IppCommand Type="SELECT" Priority="0">
        <Label>Select</Label>
        <ScreenID>2</ScreenID>
      </IppCommand>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```

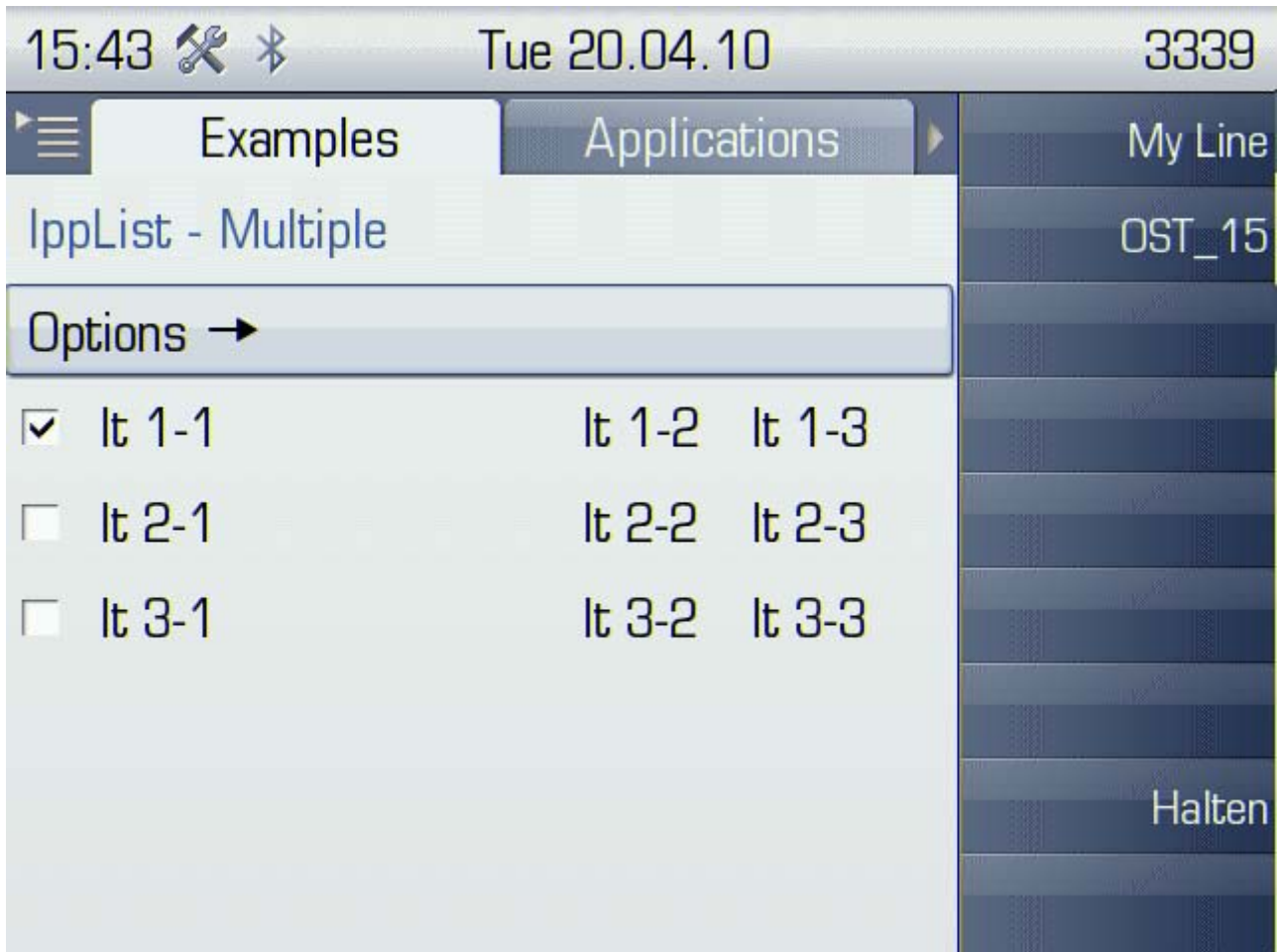

Screenshot (1 column)



Example (3 columns)

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="2" HiddenCount="0" CommandCount="2">
      <IppList Type="MULTIPLE" Count="3" Columns="3">
        <Title>IppList - Multiple</Title>
        <Url>http://${serverAddress}/testxml/servlet</Url>
        <Option ID="1" Selected="TRUE" Key="1" Value="1">
          <OptionText> It 1-1 </OptionText>
          <OptionText> It 1-2 </OptionText>
          <OptionText> It 1-3 </OptionText>
          <Image></Image>
        </Option>
        <Option ID="2" Selected="FALSE" Key="2" Value="2">
          <OptionText> It 2-1 </OptionText>
          <OptionText> It 2-2 </OptionText>
          <OptionText> It 2-3 </OptionText>
          <Image></Image>
        </Option>
        <Option ID="3" Selected="FALSE" Key="3" Value="3">
          <OptionText> It 3-1 </OptionText>
          <OptionText > It 3-2 </OptionText>
          <OptionText > It 3-3 </OptionText>
        </Option>
      </IppList>
      <IppCommand Type="SELECT" Priority="1" Key="xmlobject"
Value="_show">
        <Label>Select</Label>
        <ScreenID>1</ScreenID>
      </IppCommand>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```


Screenshot (3 columns)



5.9 Text Box

This element generates a text area for user input. For inputting data, the user must first navigate to the text box using the ▼ key, and then open the display keyboard by pressing the Ⓞ key. The use of the display keyboard is explained in the OpenStage 60/80 User Guide. When data input is completed, the user can confirm by pressing the ➔ key and the Ⓞ key, and afterwards press the ▲ key to go back to the options bar.

For a text box, predefined content, a title, the permitted data types, as well as a server URL for data transmission can be specified.

Attributes

The following attributes are available to the `IppTextBox` element:

Attribute	Value	Functionality/Remarks
MaxSize (optional)	Maximum number of characters	If missing, negative, or invalid, the maximum number of characters is limited only by the capacity of the display and the buffer of the phone.
Constraint (optional)	ANY	Default value. Any character string is permitted.
	NUMERIC	The data entered must be numeric. Other entries are ignored. The valid data range is +/- 2147483647.
	PASSWORD	The data entered are treated as a password. Asterisks (*) are displayed instead of the characters actually entered.
Password (optional)	YES	The data entered in the Text Box is confidential and should be obscured whenever possible.
	NO	Default.
Default (optional)	NULL	An empty text box is displayed; the content of the <code>Text</code> element is ignored.
	TEXT	The text box contains the content of the <code>Text</code> element.
	PHONENUMBER	The text box contains the call number of the phone.
Key (optional)	Key	The value of this attribute is used as a key for submitting the data to the server-side program. The associated value is the content of the <code>Text</code> element. Please note that the key must be unique within a Screen.
Uneditable (optional)	YES	The text box cannot be edited by the user.
	NO	Default value. The text box can be edited by the user.

Syntax

```
<IppTextBox MaxSize=" [Number] "  
Constraint="ANY | NUMERIC | PASSWORD | PHONENUMBER | URL | EMAILADDR "  
Password="YES | NO" Default="NULL | TEXT | PHONENUMBER" Key=" [Key] "  
Uneditable="YES | NO">  
  <Title> </Title>  
  <Text> </Text>  
  <Url> </Url>  
</IppTextBox>
```

Child elements

- **Title**
Title of the text box. The title is displayed in the title bar.
- **Text**
Predefined text contained in the text box. This is displayed if `Default` is set to `TEXT`.
- **Url**
URL of the server to which the data are sent.



If the URL is missing or is invalid, thereby allowing no data to be sent to the server, the current screen will continue to be displayed unchanged.

Commands

A command is required to send the text data to a server and initiate server-side action. The command is added by inserting a command object in the screen (see Example). It is displayed on the options menu, which is located underneath the screen title. For details about the command object, see Section 5.5, "Command".

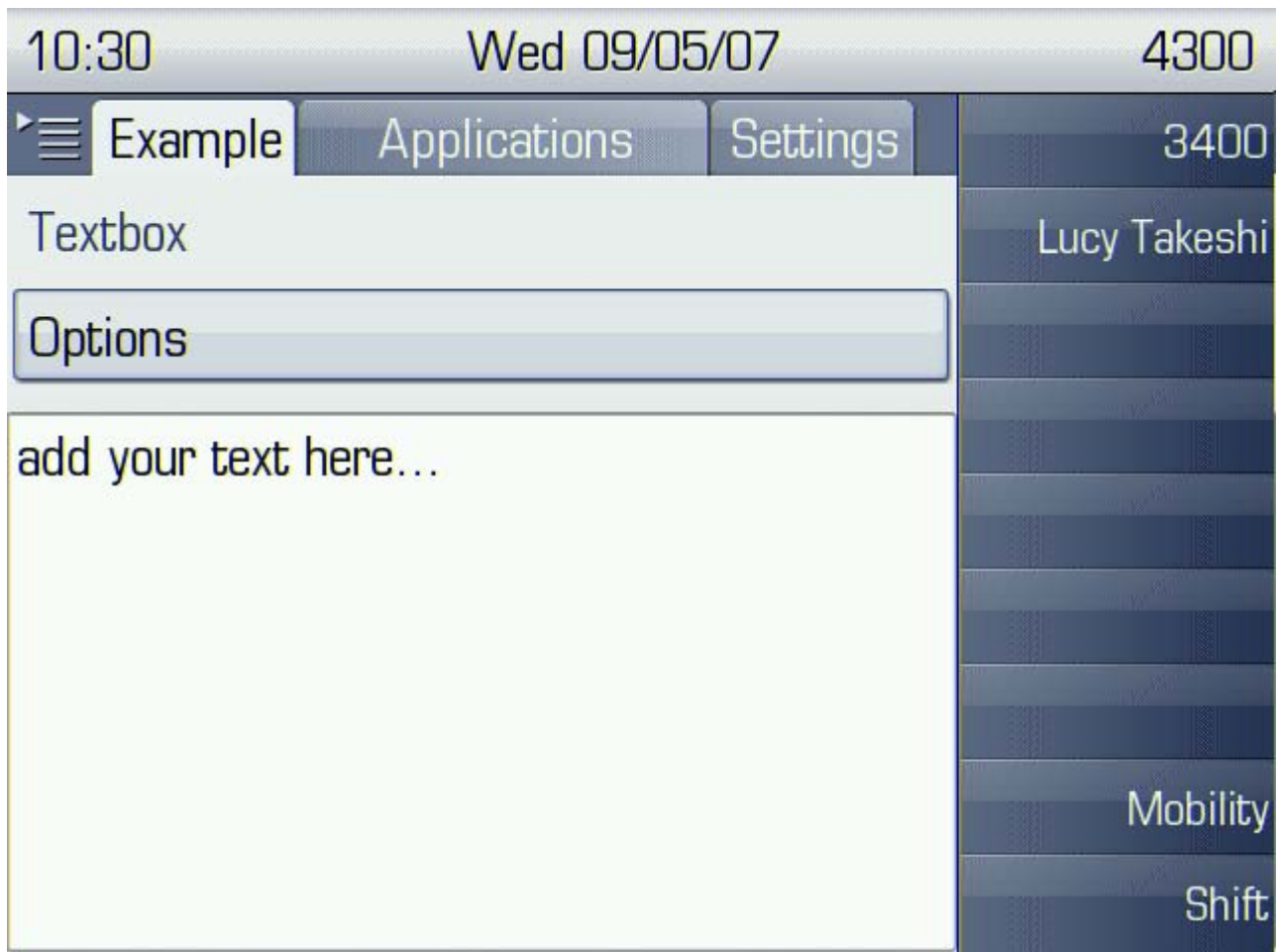
XML Object Reference

Text Box

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="1">
      <IppTextBox MaxSize="30" Constraint="ANY" Default="TEXT"
Key="TB1">
        <Title>Textbox</Title>
        <Text>add your text here...</Text>
        <Url>http://subdomain.domain/path/program</Url>
      </IppTextBox>
      <IppCommand Type="SELECT" Priority="0">
        <Label>Select</Label>
        <ScreenID>2</ScreenID>
      </IppCommand>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```


Screenshot



5.10 Ticker

Ticker text is displayed underneath the title bar and continually moves from right to left across the display.

Syntax

```
<IppTicker>
  <Text> </Text>
</IppTicker>
```

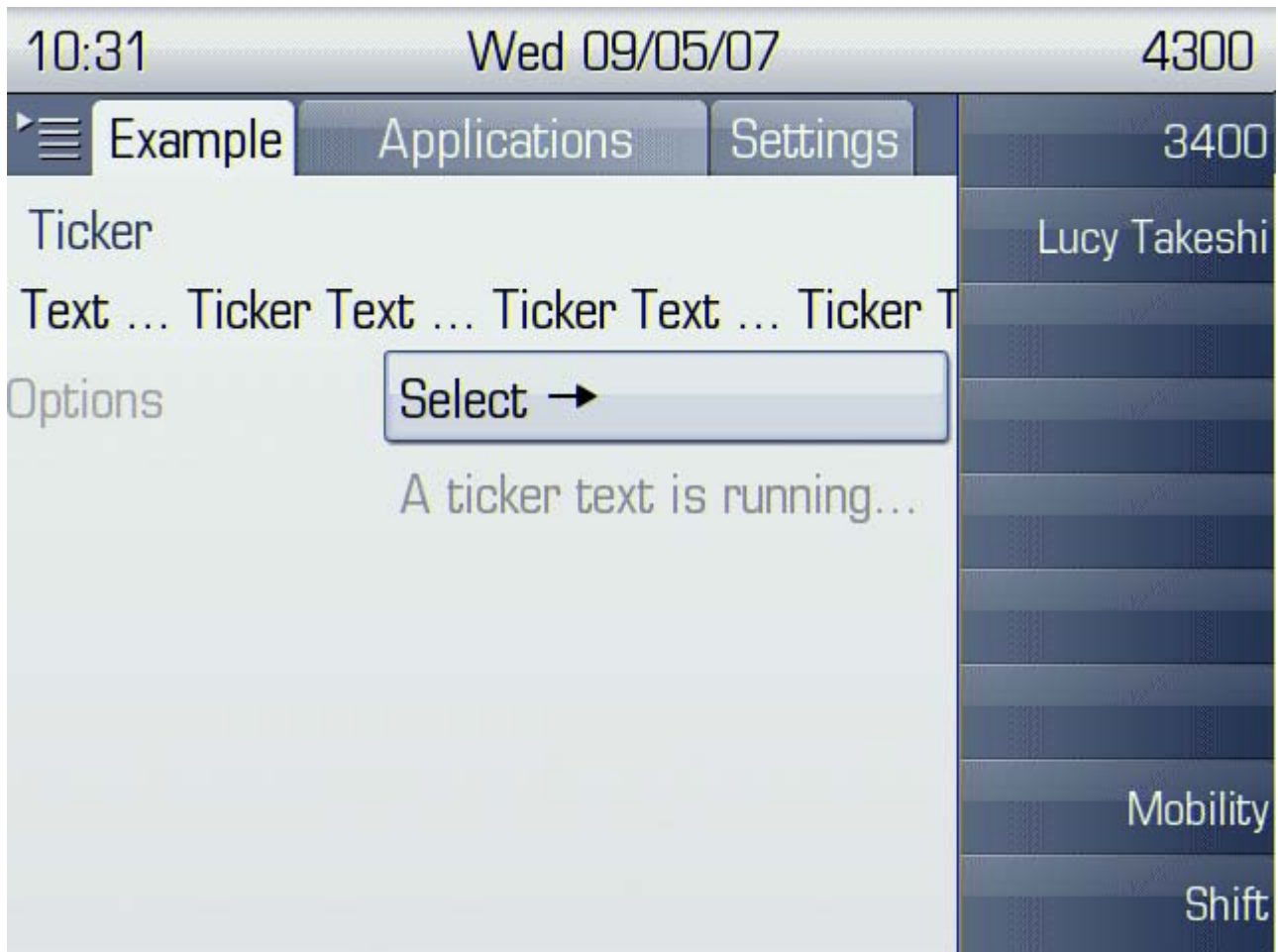


If the text is missing or is invalid, the ticker is not displayed at all.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="1">
      <IppForm ItemCount="1">
        <Title>Ticker</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <IppStringItem>
          <Label></Label>
          <Text>A ticker text is running...</Text>
        </IppStringItem>
      </IppForm>
      <IppCommand Type="SELECT" Priority="0">
        <Label>Select</Label>
        <ScreenID>1</ScreenID>
      </IppCommand>
      <IppTicker>
        <Text>Ticker Text ... Ticker Text ...</Text>
      </IppTicker>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```


Screenshot



5.11 Hidden

The function of this object is comparable to that of hidden entry fields in HTML. The object enables sending and receiving data from the server using key/value pairs, without this data being visible to the user.

Attributes

The following attributes are available to the `IppHidden` element:

Attribute	Value	Function
Type (mandatory)	VALUE	Default. The value of the key/value pair is the value saved in the <code>Value</code> child element.
	PHONENUMBER	The value of the key/value pair is the phone's call number.
	IPADDRESS	The value of the key/value pair is the IP address of the phone connected.
Key (mandatory)	Key	Together with the content of the <code>Value</code> child element, <code>Key</code> builds the key/value pair to be sent as an HTTP POST request. Please note that the key must be unique within a Screen.

Syntax

```
<IppHidden Type="VALUE | PHONENUMBER | IPADDRESS" Key=" [Key] ">  
  <Value> </Value>  
</IppHidden>
```

Child elements

- `Value`

This contains the value that is sent to the server. Together with the `Key` attribute of the `IppHidden` element, the content of `Value` builds a key/value pair.

Commands

A command is required to the hidden data to a server and initiate server-side action. The command is added by inserting a command object in the screen (see Example). can be displayed on the context menu of the options bar, or on the context menu of each individual item on a list (see Section 5.8, "List"), or both. For details about the command object, see Section 5.5, "Command".

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="1" CommandCount="1">
      <IppList Type="EXCLUSIVE" Count="3">
        <Title>Hidden</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <Option ID="1" Selected="FALSE" Key="key1" Value="1">
          <OptionText>Option 1</OptionText>
        </Option>
        <Option ID="2" Selected="FALSE" Key="key2" Value="2">
          <OptionText>Option 2</OptionText>
        </Option>
        <Option ID="3" Selected="FALSE" Key="key3" Value="3">
          <OptionText>Option 3</OptionText>
        </Option>
      </IppList>
      <IppCommand Type="SELECT" DisplayOn="LISTITEM" Priority="0">
        <Label>Select</Label>
        <ScreenID>2</ScreenID>
      </IppCommand>
      <IppHidden Type="VALUE" Key="hidden key">
        <Value>hidden value</Value>
      </IppHidden>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```


5.12 Form

The form object is used to generate a frame that provides some layout options for a screen. In the XML document, it serves as a container for displayable objects. The form provides a `Title` element for labeling the screen content and a `Url` element for sending data to the server-side program.



If objects of different types shall be combined in a form, these objects must be wrapped in an item element (see Section 5.13, "Form/Item").

The following objects can be contained in a form:

- String item (see Section 5.14, "Form/String Item")
- Image item (see Section 5.15, "Form/Image Item")
- Spacer (see Section 5.16, "Form/Spacer")
- Text ffield (see Section 5.17, "Form/Text Field")
- Choice group (see Section 5.18, "Form/Choice Group")
- Date field (see Section 5.19, "Form/Date Field")
- Button (see Section 5.20, "Form/Button")
- Gauge (see Section 5.21, "Form/Gauge")

A form is divided into two columns. The left column contains the "Options" label automatically created for `Command` elements, as well as the labels for objects within the form. The proportions of the columns can be modified using the `Proportion` attribute.

Attributes

The following attributes and values are available to the `IppForm` element:

Attribute	Value	Function
ItemCount (mandatory)	Number of objects contained	Indicates the number of specific objects contained (<code>Title</code> and <code>Url</code> are not counted).
	Missing, negative or invalid	Defaults to 0. Any objects in the form will be ignored.
Proportion (optional)	0_100	Left column: 0% - right column: 100%
	15_85	Left column: 15% - right column: 85%
	25_75	Left column: 25% - right column: 75%
	40_60	Left column: 40% - right column: 60%
	50_50	Left column: 50% - right column: 50%
	60_40	Left column: 60% - right column: 40%
	75_25	Left column: 75% - right column: 25%

Syntax

```
<IppForm ItemCount="[Number]"
Proportion="0_100|15_85|25_75|40_60|50_50|60_40|75_25">
  <Title> </Title>
  <Url> </Url>
  <IppStringItem/>
  <IppImageItem/>
  <IppSpacer/>
  <IppTextField/>
  <IppChoiceGroup/>
  <IppDateField/>
  <IppButton/>
  <IppItem>
  <IppGauge>
</IppForm>
```


XML Object Reference

Form

Child elements

- `Title`
Title for the form.
- `Url`
URL of the server to which the data is to be sent.



If the URL indicated in the `Url` element is missing or invalid, thereby allowing no data to be sent to a server, the screen will continue to be displayed unchanged.

5.13 Form/Item

By means of this element, commands can be associated with individual items in a form, as an alternative to displaying them in the options bar. When the user navigates to an item and presses the ➔ key on the TouchGuide, a context menu is opened which provides the commands.

The following objects can be contained in an Item:

- String Item (see Section 5.14, "Form/String Item")
- Image Item (see Section 5.15, "Form/Image Item")
- Spacer (see Section 5.16, "Form/Spacer")
- Text Field (see Section 5.17, "Form/Text Field")
- Choice Group (see Section 5.18, "Form/Choice Group")
- Date Field (see Section 5.19, "Form/Date Field")
- Button (see Section 5.20, "Form/Button")
- Gauge (see Section 5.21, "Form/Gauge")

Attributes

The following attribute/value combinations are available to the `IppItem` element:

Attribute	Value	Function/Remarks
CommandCount (mandatory)	Number of commands within the <code>IppItem</code>	The commands are rendered correctly.
	0	Any <code>IppCommand</code> elements will be ignored.
	Missing, negative or invalid	The value will default to 0.
	Less than the number of commands within the <code>IppItem</code>	Only those commands that fall within the given number are displayed.

Syntax

```
<IppItem CommandCount="[Number of Commands]">
  <IppStringItem/>
  <IppImageItem/>
  <IppSpacer/>
  <IppTextField/>
  <IppDateField/>
  <IppChoiceGroup/>
  <IppGauge/>
  <IppButton/>
</IppItem>
```

Commands

To add a command to an item in a form, just include an `IppCommand` in the `IppItem`. The `DisplayOn` attribute of the command must be set to `LISTITEM` or to `BOTH` (see Section 5.5, "Command"). In the latter case, all commands assigned to the items are also available from the options bar.

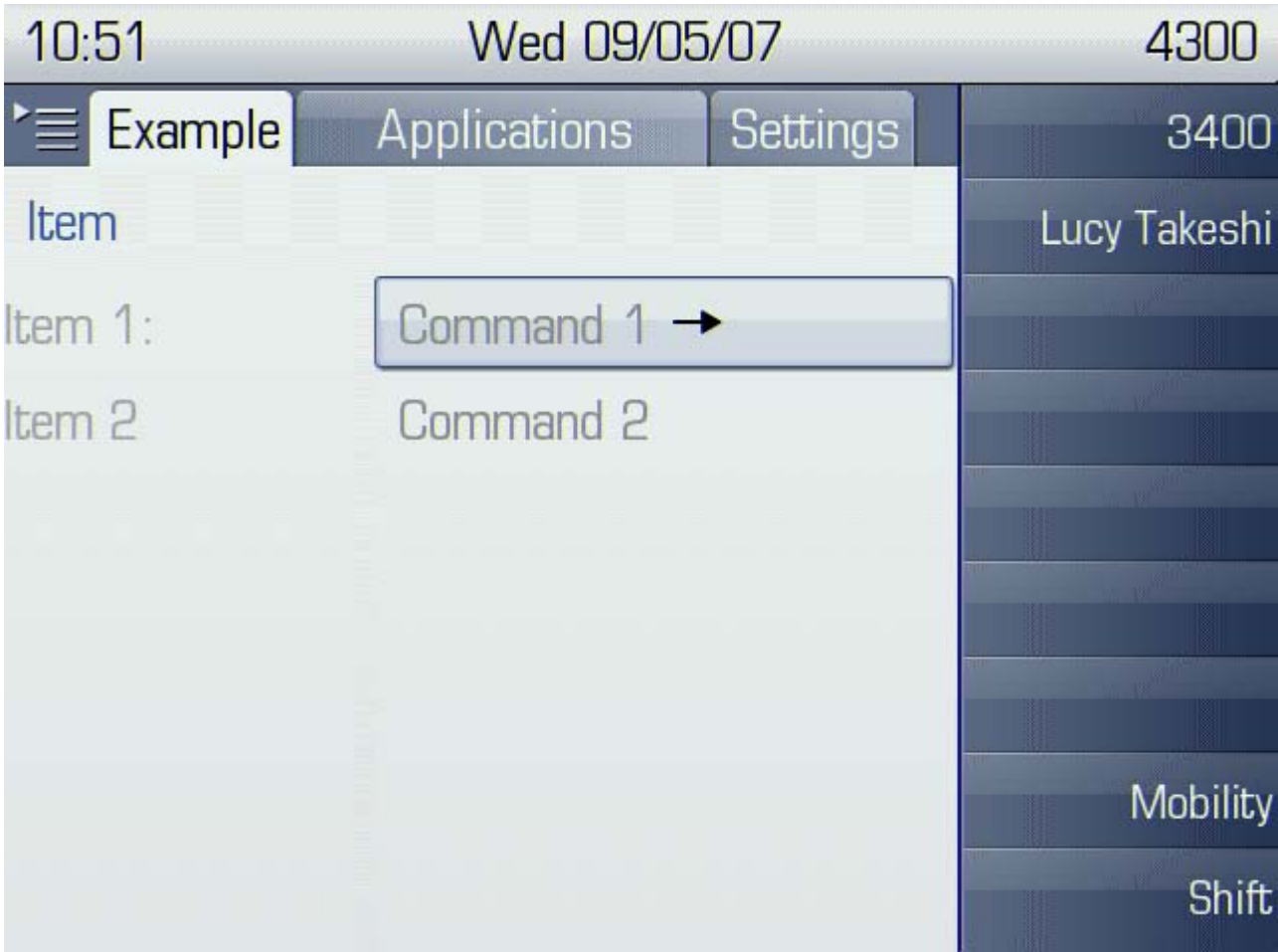
Example

```

<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="1" CommandCount="2">
      <IppForm ItemCount="2">
        <Title>Item</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <IppItem CommandCount="1">
          <IppStringItem>
            <Label>Item 1:</Label>
            <Text>Command 1</Text>
          </IppStringItem>
          <IppCommand Type="SELECT" DisplayOn="LISTITEM" Prior-
ity="0">
            <Label>Select</Label>
            <ScreenID>1</ScreenID>
          </IppCommand>
        </IppItem>
        <IppItem CommandCount="1">
          <IppStringItem>
            <Label>Item 2</Label>
            <Text>Command 2</Text>
          </IppStringItem>
          <IppCommand Type="SELECT" DisplayOn="LISTITEM" Prior-
ity="1">
            <Label>Select</Label>
            <ScreenID>1</ScreenID>
          </IppCommand>
        </IppItem>
      </IppForm>
    </IppScreen>
  </IppDisplay>
</IppPhone>

```


Screenshot



5.14 Form/String Item

This object provides a read-only text field. The text field consists of a `Label` and a `Text` element. The label is displayed in the left column, and the text is shown in the right column.

Syntax

```
<IppStringItem>
  <Label> </Label>
  <Text> </Text>
</IppStringItem>
```

Child elements

- `Label`
Label that precedes the text.
- `Text`
Content in the text field.

Commands

A string item can be made interactive by adding a command to it. To achieve this, both the string item and the command must be placed inside a special form item, which is a container for adding commands to form objects. For details, please see Section 5.13, "Form/Item". For details about the command object, see Section 5.5, "Command".

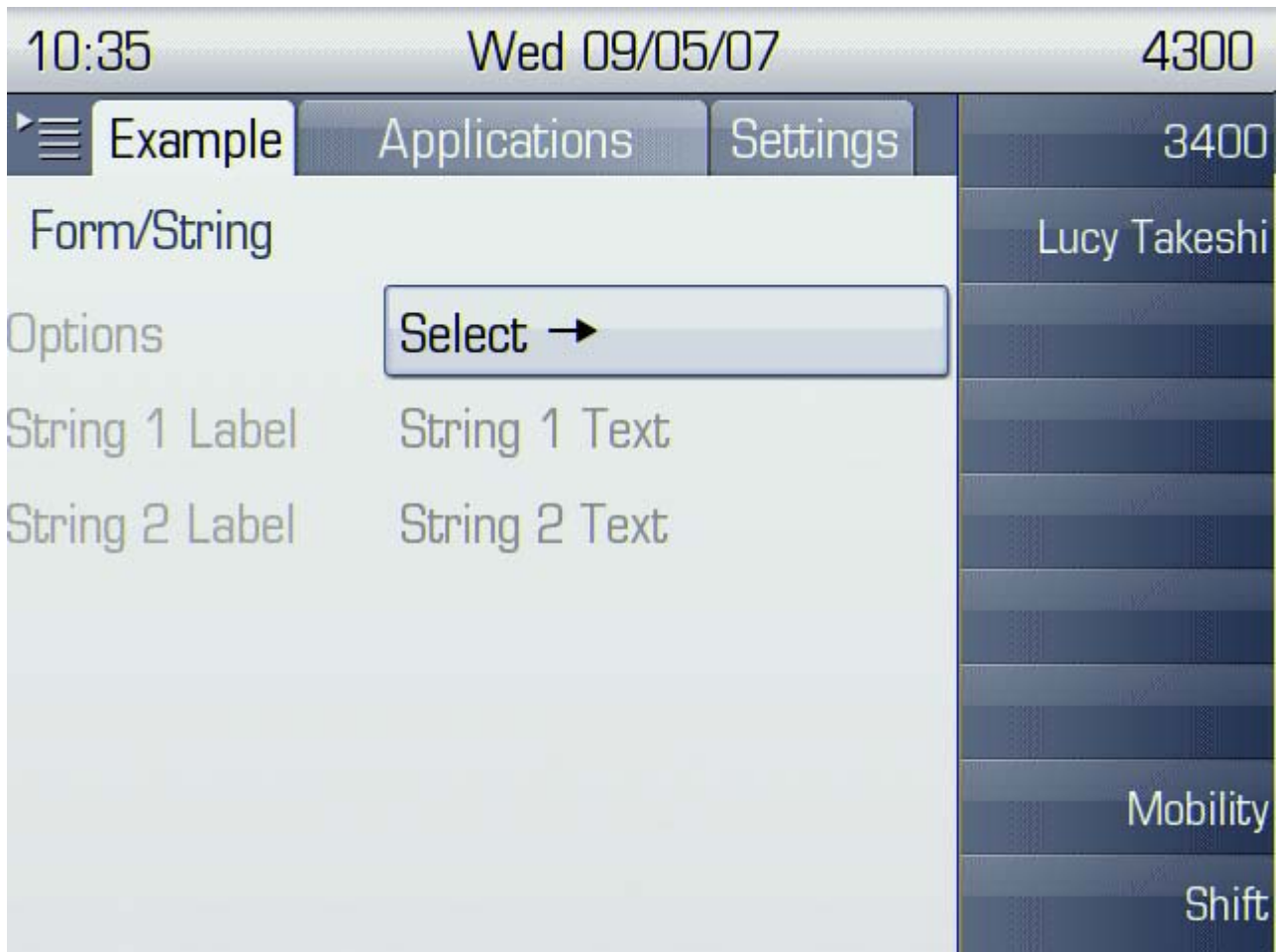
XML Object Reference

Form/String Item

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="1">
      <IppForm ItemCount="2">
        <Title>Form/String</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <IppStringItem>
          <Label>String 1 Label</Label>
          <Text>String 1 Text</Text>
        </IppStringItem>
        <IppStringItem>
          <Label>String 2 Label</Label>
          <Text>String 2 Text</Text>
        </IppStringItem>
      </IppForm>
      <IppCommand Type="SELECT" Priority="0">
        <Label>Select</Label>
        <ScreenID>2</ScreenID>
      </IppCommand>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```


Screenshot



5.15 Form/Image Item

This object functions as a frame for linking an image in the screen layout. In case the image cannot be displayed, an alternative text can be shown instead. Also, a label can be added, which is displayed in the left column.

Syntax

```
<IppImageItem>
  <Label> </Label>
  <Image Cache=" [Name] "> </Image>
  <AltText> </AltText>
</IppImageItem>
```

Child elements

- `Label`
Label for the image; displayed in the left column.
- `Image`
Links the image file. For more information about the image object, see Section 5.22, "Image".
- `AltText`
Alternative text that is displayed if the image cannot be displayed.

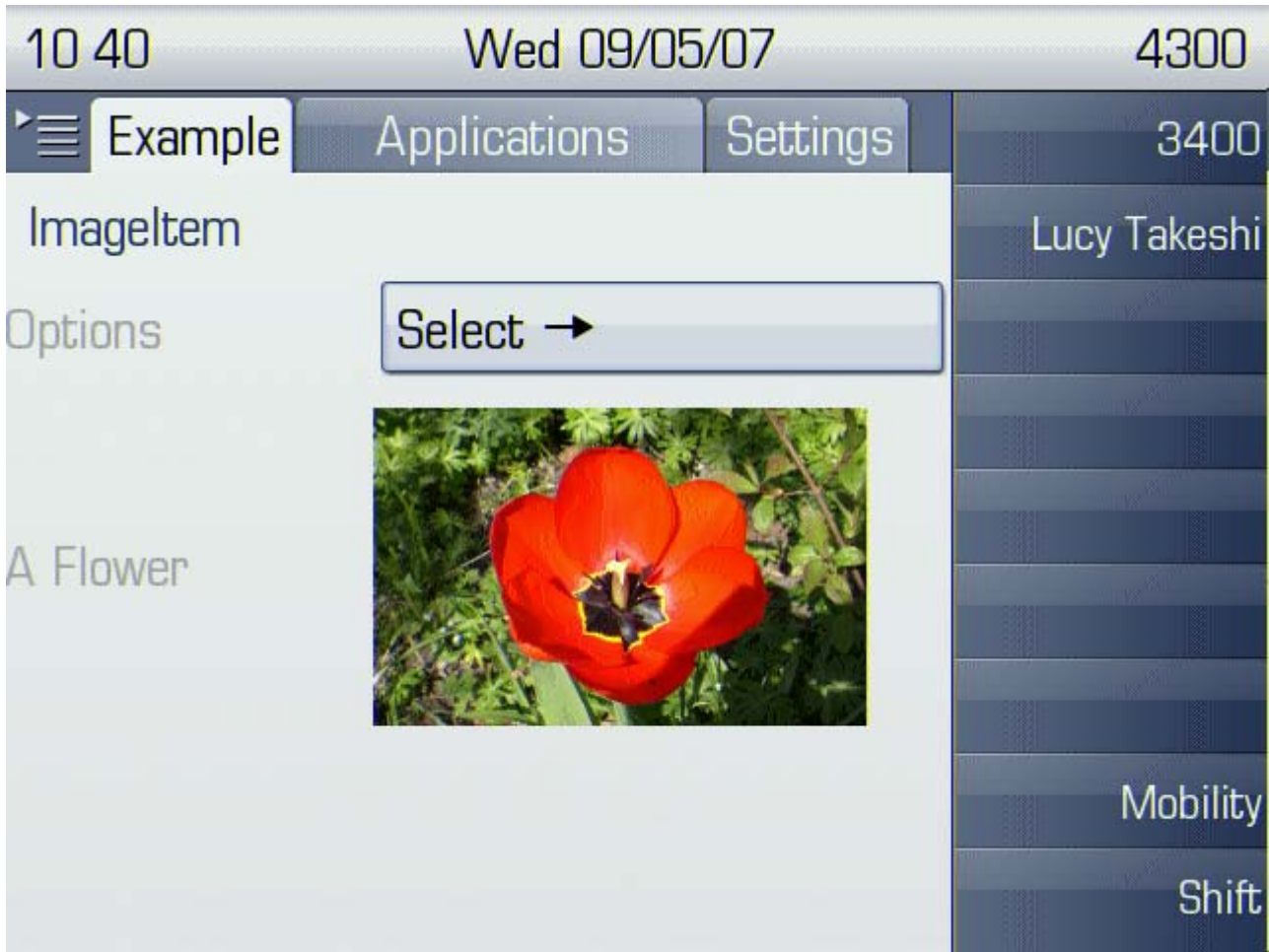
Commands

An image item can be made interactive by adding one or more commands. As a specific feature of image items, the context menu will contain an additional command for closing the context menu. This command is labeled "Exit menu". To add a command, both the image item and the command must be placed inside a form item (see Section 5.13, "Form/Item"). For details about the command object, see Section 5.5, "Command".

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="1">
      <IppForm ItemCount="1">
        <Title>ImageItem</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <IppImageItem Layout="CENTER">
          <Label>A Flower</Label>
          <Image>http://subdomain.domain/path/image.png</Image>
          <AltText>The image cannot be displayed.</AltText>
        </IppImageItem>
      </IppForm>
      <IppCommand Type="SELECT" Priority="0">
        <Label>Select</Label>
        <ScreenID>2</ScreenID>
      </IppCommand>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```


Screenshot



5.16 Form/Spacer

The spacer generates an extra vertical space the height of a line between two elements.



As inserting a spacer between two objects implies introducing a different type of object in the XML document, all form objects, including the spacer itself, must be wrapped in a form item element (see Section 5.13, "Form/Item").

Attributes

For the `IppSpacer` element, the `NewLine` attribute is required with the following values:

Attribute	Value	Functionality/Remarks
NewLine (mandatory)	NEWLINE_AFTER	Not implemented yet.
	NEWLINE_BEFORE	Not implemented yet.
	NEWLINE_BEFO_AFT	Not implemented yet.

Syntax

```
<IppSpacer NewLine="NEWLINE_AFTER|NEWLINE_BEFORE|NEWLINE_BEFO_AFT" />
```

Commands

A command can be added to a spacer. To achieve this, both the spacer and the command must be placed inside a special form item, which is a container for adding commands to form objects. For details, please see Section 5.13, "Form/Item". For details about the command object, see Section 5.5, "Command".



When a command is associated with a spacer by means of a form item, the `DisplayOn` attribute of the form must be set to `OPTIONS`.

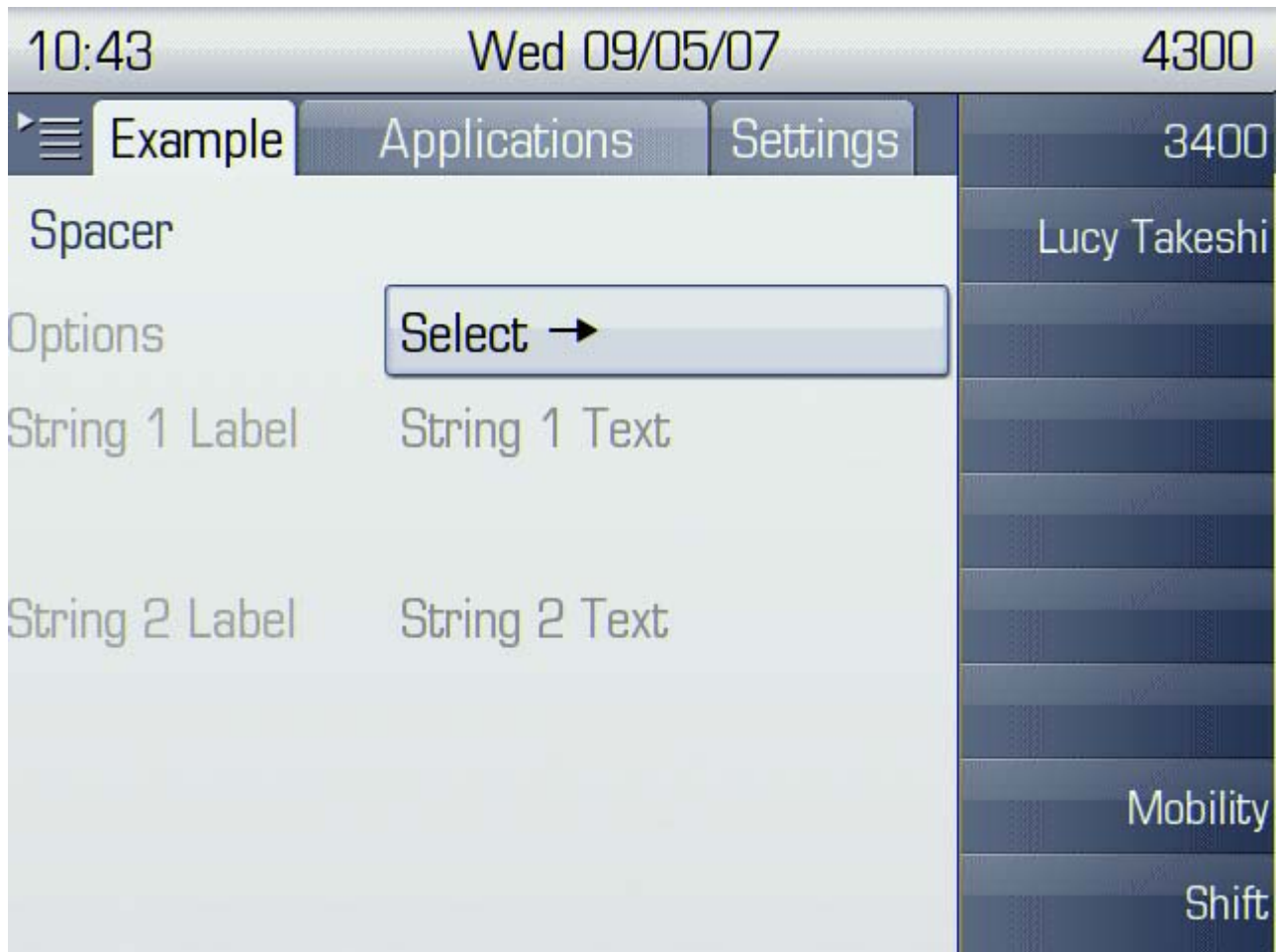
XML Object Reference

Form/Spacer

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="1">
      <IppForm ItemCount="3">
        <Title>Spacer</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <IppItem>
          <IppStringItem>
            <Label>String 1 Label</Label>
            <Text>String 1 Text</Text>
          </IppStringItem>
        </IppItem>
        <IppItem>
          <IppSpacer/>
        </IppItem>
        <IppItem>
          <IppStringItem>
            <Label>String 2 Label</Label>
            <Text>String 2 Text</Text>
          </IppStringItem>
        </IppItem>
      </IppForm>
      <IppCommand Type="SELECT" Priority="0">
        <Label>Select</Label>
        <ScreenID>2</ScreenID>
      </IppCommand>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```


Screenshot



5.17 Form/Text Field

An entry field for user provided data is generated, similar to the `<input>` tag in HTML. The entry field contains a label. In addition, the data type of the entry can be defined using the `Constraint` attribute.

Attributes

The following attributes are available to the `TextField` element:

Attribute	Value	Function/Remarks
MaxSize (optional)	Maximum number of characters	Data input is accepted until this number is reached. Any further characters will be ignored.
	Missing, negative, or invalid	The maximum number of characters is limited only by the capacity of the display and the buffer of the phone
Constraint (optional)	ANY	Default value. Any character string is permitted.
	NUMERIC	The data entered must be numeric. Other keyboard entries are ignored.
	PASSWORD	The data entered are treated as a password. Asterisks (*) are displayed instead of the characters actually entered.
	PHONENUMBER	The data entered must be in a telephone number format.
	URL	The data entered must represent a valid URL.
	EMAILADDR	The data entered must represent a valid e-mail address.
PASSWORD (optional)	YES	The data entered in the text field are confidential and should be obscured whenever possible.
	NO	Default value. The data entered in the text field are not confidential.
Default (optional)	NULL	An empty entry field is displayed; the content of the <code>Text</code> element is ignored.
	TEXT	The entry field contains the content of the <code>Text</code> element.
	PHONENUMBER	The text field contains the phone's call number.

Table 5-2

Attribute	Value	Function/Remarks
Key (optional)	Key	The value of this attribute is used as a key for submitting the data as an HTTP request. The associated value is the content of the <code>Text</code> element. Please note that the key must be unique within a screen.
Uneditable (optional)	YES	The text field cannot be edited by the user.
	NO	Default value. The text field can be edited by the user.

Table 5-2

Syntax

```
<IppTextField MaxSize="[Number]" Constraint="ANY|NUMERIC|
PASSWORD|PHONENUMBER|URL|EMAILADDR" PASSWORD="YES|NO"
Default="NULL|TEXT|PHONENUMBER" Key="[Key]" Uneditable="YES|NO">
  <Label > </Label >
  <Text> </Text>
</IppTextField >
```

Child elements

- Label

Label for identifying the entry field. It is displayed in a two-column layout; the label is placed in the left column, and the entry field is placed in the right column.

- Text

Preset text in the entry field.

Commands

To send the data to a server and initiate server action, an appropriate command must be added to the screen (see Section 5.5, "Command"). The command will be displayed on the context menu of the options bar. Alternatively, the command can be associated with a text field. To achieve this, both the text field and the command must be placed inside a form item. For details, please see Section 5.13, "Form/Item".



If a command is associated with the text field, an arrow is shown as soon as the field has the focus. Additionally, the TouchGuide buttons are mapped in a different way: the → key now opens the context menu, so that only the ⓘ key opens the editor.

XML Object Reference

Form/Text Field

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="1">
      <IppForm ItemCount="3">
        <Title>Form String</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <IppTextField MaxSize="30" Constraint="NUMERIC" De-
fault="TEXT" Key="TF1">
          <Label>Numeric Text</Label>
          <Text>123456</Text>
        </IppTextField>
        <IppTextField MaxSize="30" Constraint="ANY" De-
fault="TEXT" Key="TF2">
          <Label>Any Text</Label>
          <Text>abc123</Text>
        </IppTextField>
        <IppTextField MaxSize="30" Constraint="PASSWORD" De-
fault="TEXT" Key="TF3">
          <Label>Password</Label>
          <Text>999</Text>
        </IppTextField>
      </IppForm>
      <IppCommand Type="SELECT" Priority="0">
        <Label>Select</Label>
        <ScreenID>2</ScreenID>
      </IppCommand>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```


Screenshot

The screenshot displays a mobile application interface. At the top, a status bar shows the time '10 45', the date 'Wed 09/05/07', and the battery level '4300'. Below the status bar is a navigation bar with three tabs: 'Example' (selected), 'Applications', and 'Settings'. The main content area is divided into two sections. The left section contains a list of form fields: 'Form String', 'Options', 'Numeric Text', 'Any Text', and 'Password'. The right section contains a list of items: '3400', 'Lucy Takeshi', and 'Mobility'. The 'Options' field is currently selected, and a dropdown menu is visible with the text 'Select →'. The 'Numeric Text' field contains the value '123456', the 'Any Text' field contains 'abc123', and the 'Password' field contains '***'. The '3400' item is highlighted in blue, and the 'Mobility' item is also highlighted in blue. The 'Shift' item is visible at the bottom of the list.

Form Field	Value
Form String	
Options	Select →
Numeric Text	123456
Any Text	abc123
Password	***

Item
3400
Lucy Takeshi
Mobility
Shift

5.18 Form/Choice Group

This object generates a selection list within a form, comparable to the `<select>` and `<option>` elements in HTML and WML.

A form may contain none, one, or more choice groups.

Attributes

The following attributes are available to the `IppChoiceGroup` element:

Attribute	Value	Function
Type (mandatory)	EXCLUSIVE	Default value. Exactly one option can be selected (radio button function). Only the selected key/value pair is sent to the server.
	MULTIPLE	One or more options can be selected. All the key/value pairs are sent to the server. For selected options, the value of the <code>value</code> attribute is sent; for non-selected options, this value is sent with the prefix <code>not_</code> .
	POPUP	On startup, only the preselected option is shown. On pressing → , a context menu containing all options opens up. A context menu option is selected by pressing Ⓚ when it has the focus. To submit the selection made in the context menu, the <code>SELECT</code> command must be issued from the options bar, just like with the other choice group types.
Count (mandatory)	Number of options	Only as many options as indicated here are displayed. If the actual number of options is greater, all further options that exceed this setting are ignored.
	0	No options are displayed.

Syntax

```
<IppChoiceGroup Type="EXCLUSIVE|MULTIPLE|POPUP" Count="[Number]">
  <Label> </Label>
  <Option Selected="FALSE|TRUE" Key="[Key]" Value="[Value]">
    <OptionText> </OptionText>
    <IppPhoneNumber> </IppPhoneNumber>
    <Image Cache="[Name]"> </Image>
  </Option>
</IppChoiceGroup>
```


Child elements

- Option

Stores the data to be sent to the server in attributes. It can also contain descriptive text (`OptionText`) and an image (`Image`).

Attribute	Value	Function
Selected (optional)	FALSE	Default value. The option is not preselected
	TRUE	The option is preselected.
Key (optional)	Key	The key for the key/value pair that is sent to the server. For selection lists of type <code>POPUP</code> and <code>EXCLUSIVE</code> , multiple options can have the same <code>Key</code> . If the type is <code>MULTIPLE</code> , all the values must be different.
Value (optional)	Value	The value for the key/value pair that is sent to the server. If the selection list type is <code>MULTIPLE</code> , all the key/value pairs are sent. For selected options, the value of the <code>value</code> attribute is then sent; for non-selected options, this value is sent with the prefix <code>not_..</code>

Table 5-3



If a key or value is missing or invalid, `null` is sent instead.



The uniqueness of keys and values is not checked in the telephone

- OptionText

Descriptive text for the option. An `IppPhoneNumber` (see Section 5.24, "Phone Number") can be contained here. In this case, the corresponding call number or the associated name can be displayed here, if the call number is stored in the local phone book.

- Image

Illustrative image for the option. The image is displayed in the left column. For more information about the `Image` object, see Section 5.22, "Image".

Commands

To send the data to a server and initiate server action, an appropriate command must be added to the screen (see Section 5.5, "Command"). The command will be displayed on the context menu of the options bar. Alternatively, a command can be associated with the choice group by means of a form item (see Section 5.13, "Form/Item").

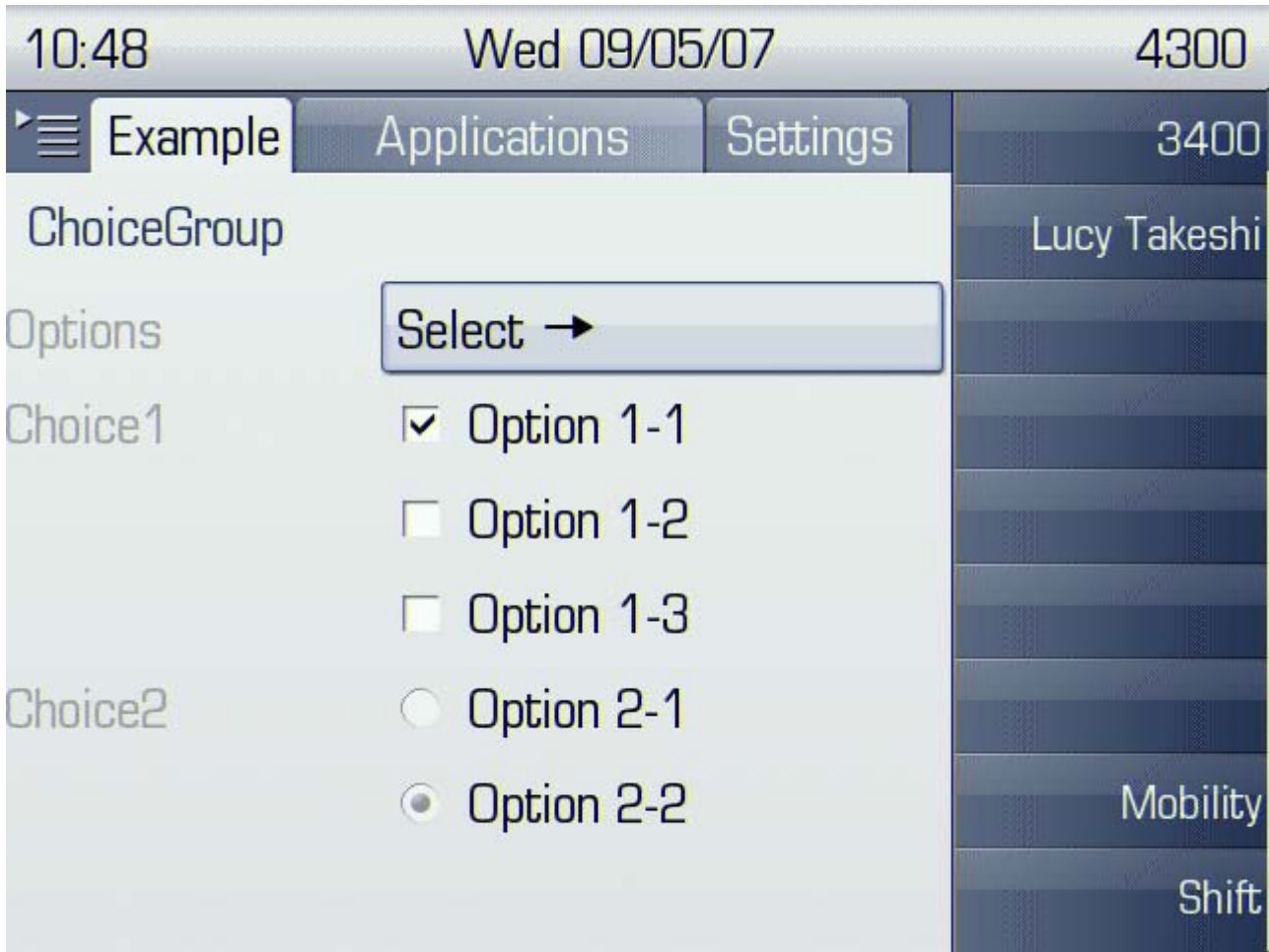
Example

```

<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="1">
      <IppForm ItemCount="2">
        <Title>ChoiceGroup</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <IppChoiceGroup Type="MULTIPLE" Count="3">
          <Label>Choice1</Label>
          <Option Selected="TRUE" Key="key1-1" Value="1">
            <OptionText>Option 1-1</OptionText>
            <Image></Image>
          </Option>
          <Option Selected="FALSE" Key="key1-2" Value="2">
            <OptionText>Option 1-2</OptionText>
            <Image></Image>
          </Option>
          <Option Selected="FALSE" Key="key1-3" Value="3">
            <OptionText>Option 1-3</OptionText>
            <Image></Image>
          </Option>
        </IppChoiceGroup>
        <IppChoiceGroup Type="EXCLUSIVE" Count="2">
          <Label>Choice2</Label>
          <Option Selected="FALSE" Key="key2-1" Value="4">
            <OptionText>Option 2-1</OptionText>
            <Image></Image>
          </Option>
          <Option Selected="TRUE" Key="key2-2" Value="5">
            <OptionText>Option 2-2</OptionText>
            <Image></Image>
          </Option>
        </IppChoiceGroup>
      </IppForm>
      <IppCommand Type="SELECT" Priority="0">
        <Label>Select</Label>
        <ScreenID>2</ScreenID>
      </IppCommand>
    </IppScreen>
  </IppDisplay>
</IppPhone>

```


Screenshot



5.19 Form/Date Field

This object generates a field in which date and time information can be preset or entered by the user. The data to be displayed (date, time, or date and time), the time zone, and a label can be specified.

The date is shown as three separate fields: day, month, and year. The time is shown as two separate fields: hours and minutes. The user cannot enter seconds; however, seconds are returned with a default value of 00.000



If the key in a date field is invalid or not present, the URL indicated in the form is opened, but the data contained in the date field are not sent.

Attributes

The following attributes are available to the `IppDateField` element:

Attribute	Value	Function
Mode (optional)	DATE	Only the content of the <code>Date</code> field is submitted to the server.
	TIME	Only the content of the <code>Time</code> field is submitted to the server.
	DATETIME	Default value. The contents of the <code>Date</code> field and of the <code>Time</code> field are submitted to the server.
Default (optional)	NULL	Default value. The field is empty when the XML document is loaded.
	MODE	The contents of <code>Date</code> or <code>Time</code> are displayed when the XML document is loaded.
DateKey (optional)	Key for the date field	Together with the content of the <code>Date</code> field, this constitutes the key/value pair to be sent to the server.
TimeKey (optional)	Key for the time field	Together with the content of the <code>Time</code> field, this constitutes the key/value pair to be sent to the server.

Syntax

```
<IppDateField Mode="DATE|TIME|DATETIME" Default="NULL|MODE"  
DateKey=" [Key] " TimeKey=" [Key] ">  
  <Label> </Label>  
  <TimeZone> </TimeZone>  
  <Date> </Date>  
  <Time> </Time>  
</IppDateField>
```

Child elements

- Label

Generates a label for identifying the date and time fields. The label is displayed in the left hand column.

- TimeZone

Determines the time zone corresponding to the date and time fields. This affects the displayed time when the `Time` element is not provided in the XML document, so that the phone's local or GMT time is displayed. Any time zone definition accepted by Java can be used here. If the content is missing or invalid, it will default to `GMT`.

Example: If the time on the phone is `13:00:00.000`, and the time zone is set to `GMT+2`, the phone's display will read `15:00`.

- Date

Entry field for the date. The format for entry and display is set in the user menu, under **Locality > Date format**.



For predefining the date in the XML document, the `YYYY-MM-DD` format is used.

Example: `2007-11-20`.

- Time

Time data is entered in the order of hours - minutes.



For predefining the time in the XML document, the `HH-MM-SS` format is used, followed by a period and the milliseconds.

Example: `13:00:00.000`.

Commands

To send the data to a server and initiate server action, an appropriate command must be added to the screen (see Section 5.5, "Command"). The command will be displayed on the context menu of the options bar.



The `DisplayItem` attribute of the command must be set to `LISTITEM` or `BOTH`; otherwise, the command will not be accessible to the user. However, it is possible to have a command displayed in the context menu of the date field by means of a form item (see Section 5.13, "Form/Item").

XML Object Reference

Form/Date Field

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="1">
      <IppForm ItemCount="3">
        <Title>DateField</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <IppDateField Mode="TIME" Default="MODE" DateKey="DK1 "
TimeKey="TK1">
          <Label>Time</Label>
          <TimeZone>GMT+1</TimeZone>
          <Time>13:00:00.000</Time>
        </IppDateField>
        <IppDateField Mode="DATE" Default="MODE" DateKey="DK2 "
TimeKey="TK2">
          <Label>Date</Label>
          <TimeZone>GMT + 1</TimeZone>
          <Date>2007-09-03</Date>
        </IppDateField>
        <IppDateField Mode="DATETIME" Default="MODE"
DateKey="DK3" TimeKey="TK3">
          <Label>Date and Time</Label>
          <TimeZone>GMT + 1</TimeZone>
          <Date>2007-09-03</Date>
          <Time>13:00:00.000</Time>
        </IppDateField>
      </IppForm>
      <IppCommand Type="SELECT" Priority="0">
        <Label>Select</Label>
        <ScreenID>2</ScreenID>
      </IppCommand>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```


Screenshot

The screenshot displays a mobile application interface. At the top, a status bar shows the time '10:51', the date 'Wed 09/05/07', and a value '4300'. Below this is a navigation bar with three tabs: 'Example' (selected), 'Applications', and 'Settings'. To the right of the tabs is a vertical list of values: '3400', 'Lucy Takeshi', and 'Mobility'. The main content area is divided into two sections. The left section, titled 'DateField', contains a 'Select' button with a right-pointing arrow. Below this are three rows of data: 'Time' with the value '13:00', 'Date' with the value '09/03/2007', and 'Date and Time' with the value '09/03/2007 13:00'. The right section of the main content area contains a vertical list of values: 'Shift', 'Mobility', and 'Shift'.

Time	Date	Date and Time
13:00	09/03/2007	09/03/2007 13:00

5.20 Form/Button

This object generates a clickable area with a label that can be specified. When clicked, the key/value pair contained in the attributes of the `IppButton` element as well as the key/value pairs of any hidden objects in the XML document (see Section 5.11, "Hidden") are immediately sent to the server. The URL of the server-side program is indicated in the form (see Section 5.12, "Form").

Adjacent buttons are placed vertically.

Attributes

The following attributes are available to the `IppButton` element:

Attribute	Value	Function
Type (optional)	IMAGE	Default value. In the current versions, this is the only possible value. The button is represented by an image.
Key (mandatory)	Key	The key for the key/value pair that is sent to the server. Please note that the key must be unique within a screen.
Value (mandatory)	Value	The value for the key/value pair that is sent to the server.

Table 5-4

Syntax

```
<IppButton Type="IMAGE" Key="[Key]" Value="[Value]">  
  <Label> </Label>  
  <Image Cache="n"> </Image>  
</IppButton>
```

Child elements

- Label
Generates a label for identifying the button.
- Image
Links an image file (see Section 5.22, "Image"). The image is rendered as a clickable area.

Commands

If an additional command is added to a button using a form item (see Section 5.13, "Form/Item"), it will displayed in the context menu of the options bar.

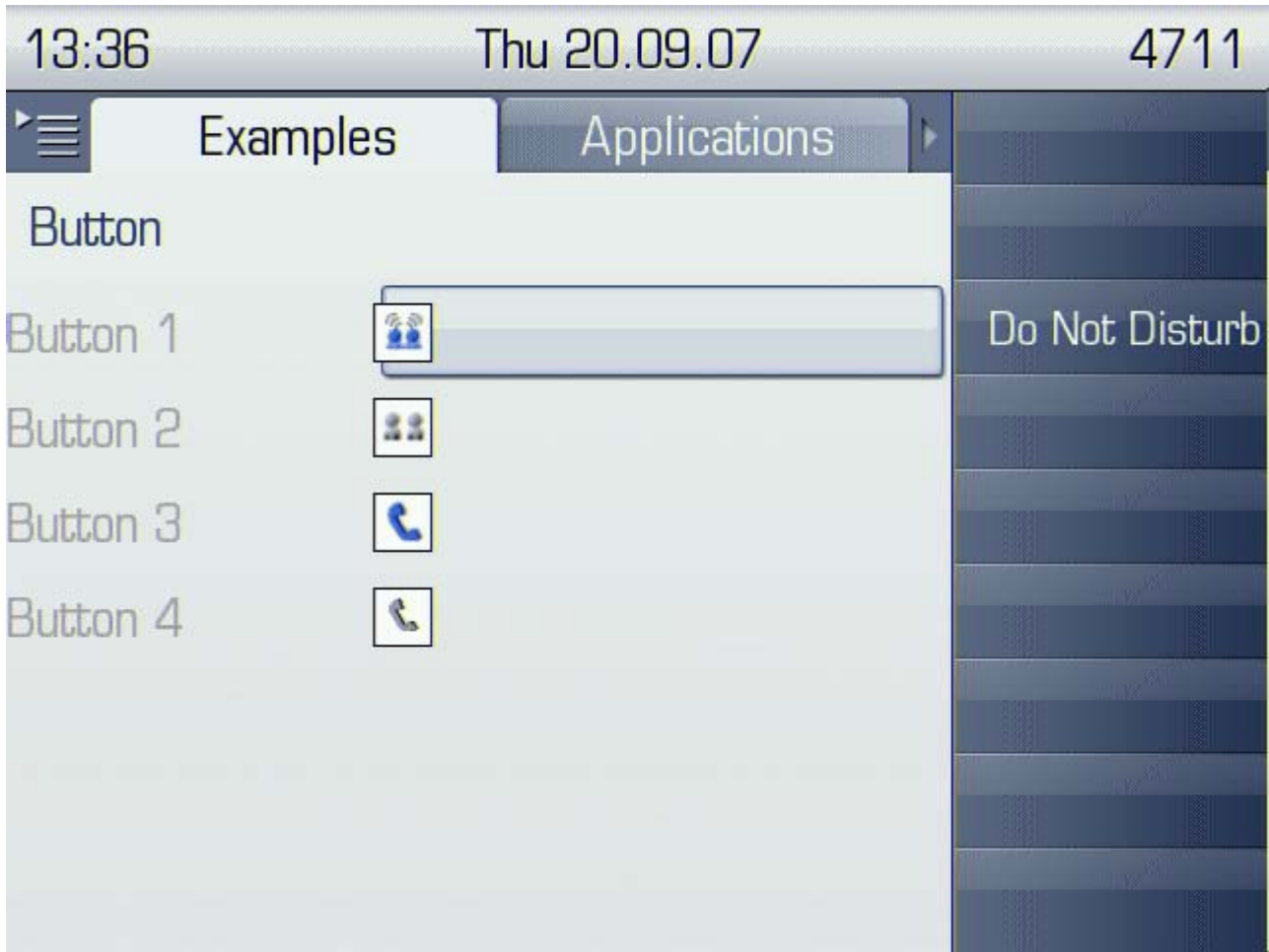
Example

```

<?xml version="1.0" encoding="UTF-8"?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="0">
      <IppForm ItemCount="4">
        <Title>Button</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <IppButton Type="IMAGE" Layout="CENTER" Key="key1 "
Value="1">
          <Label>Button 1</Label>
          <Image>http://subdomain.domain/path/image1.png</Im-
age>
        </IppButton>
        <IppButton Type="IMAGE" Layout="CENTER" Key="key2 "
Value="2">
          <Label>Button 2</Label>
          <Image>http://subdomain.domain/path/image2.png</Im-
age>
        </IppButton>
        <IppButton Type="IMAGE" Layout="CENTER" Key="key3 "
Value="3">
          <Label>Button 3</Label>
          <Image>http://subdomain.domain/path/image3.png</Im-
age>
        </IppButton>
        <IppButton Type="IMAGE" Layout="CENTER" Key="key4 "
Value="4">
          <Label>Button 4</Label>
          <Image>http://subdomain.domain/path/image4.png</Im-
age>
        </IppButton>
      </IppForm>
    </IppScreen>
  </IppDisplay>
</IppPhone>

```








Screenshot



5.21 Form/Gauge

This object creates a horizontal graphical bar which represents a value or time interval. The associated label can be used for providing information on the specific function of the gauge.

Depending on what is specified, the gauge can function as slider or as progress bar. With the slider, the user can adjust a value via the TouchGuide and send it to the server-side program. The progress bar is a horizontal bar which grows from the left to the right; the speed is specified in the XML document.

For setting a value, the user must first navigate to the gauge and then activate it by pressing the  or the  key. The slider is moved by pressing the  or  key, or by rotating the finger over the TouchGuide. To confirm the setting, the user must press the  key and afterwards press the  key to go back to the options bar.

Attributes

The following attributes are available to the `IppGauge` element:

Attribute	Value	Function
Interactive (optional)	USER	Default value. The user can change the value using the TouchGuide; the changes are graphically represented by the bar. The interactive gauge has a value range between 0 and 100 which is pre-defined by the <code>Maximum</code> and the <code>Minimum</code> element.
	AUTO	The user has no influence on the graphical bar. The gauge increments every second. The range, and, consequently, the time interval, is specified by the content of the <code>Maximum</code> element and by the content of the <code>Initial</code> element.
Key (optional)	Key for the current value	Only effective if the value of the <code>Interactive</code> attribute is <code>USER</code> . Together with the value currently set (value range 1-100), it forms a key/value pair that can be sent to the server-side program. If the key is invalid or not present, the specified URL is opened, but no data are sent to the server. For sending data to the server, the screen must contain a <code>SELECT</code> command. Please note that the key must be unique within a screen.

Syntax

```
<IppGauge Interactive="USER|AUTO" Key="k">  
  <Label> </Label>  
  <Maximum> </Maximum>  
  <Initial> </Initial>  
</IppGauge>
```

Child elements

- `Label`

Provides information on what is displayed by the gauge.

- `Initial`

Data format and function are differentiated depending on the value of the `Interactive` attribute:

- If `INTERACTIVE="USER"`: The number indicated here determines the initial value and, therefore, the starting position of the slider bar. The range is 0-100. If the value is 20, for instance, the progress bar is displayed at 20% of its total width. If the value is 0, the bar is on the leftmost position.
- If `INTERACTIVE="AUTO"`: The number indicated here determines the starting position of the progress bar. The value is given in mm:ss or in ss format.



If the initial value is missing, negative, or invalid, it will default to 0.

- Maximum

Data format and function are differentiated depending on the value of the `Interactive` attribute:

- If `INTERACTIVE="USER"`: The number entered here indicates the maximum value that can be sent to the server. The range is 0-100.



If the maximum value is set at under 100, the slider bar will still reach to the right end of the scale. If, for instance, `Maximum` is set to 80, and the user sets the slider to rightmost position, the value 80 will be sent to the server.



If the maximum value is missing, negative, invalid, or greater than 100, it will default to 100. If the initial value is greater than the maximum value, it will be set to the maximum value.

- If `INTERACTIVE="AUTO"`: If the type is non-interactive, the time duration is indicated until the progress bar has reached the right end. The value is given in mm:ss or in ss format, like, for instance, `01:20` for one minute and 20 seconds. The range is not limited.





If the maximum value is missing, negative, or invalid, it will default to 100 seconds. If the initial value is greater than the maximum value, it will be set to the maximum value and there will be no automatic updates to the gauge.

Commands

If `INTERACTIVE` is set to `USER`, data can be sent to a server and initiate server action. For this purpose, an appropriate command must be added to the screen (see Section 5.5, "Command") or, alternatively, associated with the Gauge by means of a form item (see Section 5.13, "Form/Item"). If the command is added to the screen, it will be displayed on the context menu of the options bar. If it is placed in a form item, the command will be displayed on the context menu of the Gauge.



If a Command is displayed on the Gauge, the TouchGuide buttons are mapped in a different way: First, the user must press the  key in order to change the value; after this, the  key opens the context menu.

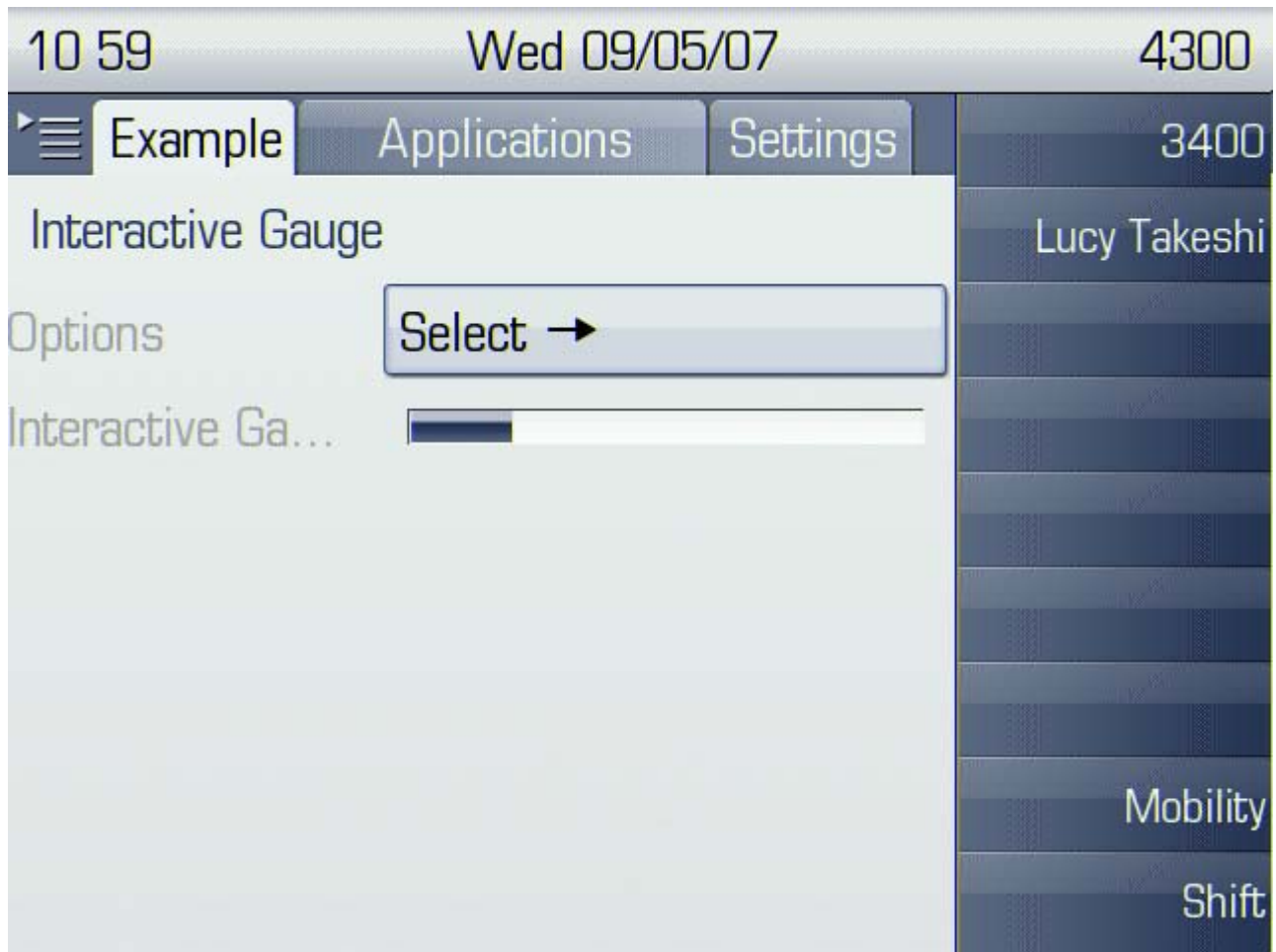
XML Object Reference

Form/Gauge

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<IppPhone>
  <IppDisplay>
    <IppScreen ID="1" HiddenCount="0" CommandCount="1">
      <IppForm ItemCount="1">
        <Title>Interactive Gauge</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <IppGauge Interactive="USER" Key="G1">
          <Label>Interactive Gauge</Label>
          <Maximum>100</Maximum>
          <Initial>20</Initial>
        </IppGauge>
      </IppForm>
      <IppCommand Type="SELECT" Priority="0">
        <Label>Select</Label>
        <ScreenID>2</ScreenID>
      </IppCommand>
    </IppScreen>
  </IppDisplay>
</IppPhone>
```


Screenshot



5.22 Image

This object is used to link an image. The XML document specifies whether the image should be cached or not. The image is downloaded from the server via HTTP, analogous to images in HTML pages.



Saving images in cache is explicitly indicated in the XML document. For this purpose, the cache name must be defined. This is necessary because specific areas in the cache are overwritten when new images are loaded while memory is on the decline. When this occurs, the previously cached images that have been overwritten will have to be reloaded if needed again.

Attributes

The following attributes are available to the `Image` element:

Attribute	Value	Function
Cache (optional)	Name	The image can be accessed from the cache under the name indicated here. If the attribute is invalid or not present, it is not cached.

Syntax

```
<Image Cache=" [Name] ">[source]</Image>
```

The content of the image element indicates the source path and filename of the image to be displayed. Depending on the situation, this path may have various formats:

The source path of an image may take one of the following formats:

- URL for the image file: `http://<IP address>:<Port>/<filename>`
- URL of a program on the server that sends the desired image file; the key/value pair identifies the image:
`http://<IP address>:<Port>/<filename of the program>?<key/value pair>`

If the image has been stored in cache already, it is sufficient to provide the file name, as in this example:

```
<Image Cache="Flower">flower.png</Image>
```



If the source of the image file is missing, but the cache attribute exists and the image is cached, the image in cache is displayed.

Images as Icons

If an image is to be used as an icon for an interactive element, it is recommended to choose an appropriate size. The images should be in PNG format. The built-in icons for OpenStage 60/80 are sized as follows:

- Icons for mode keys, call states, phone number identification, presence, media widget, speller, idle screen, info bar, and others: 36px x 36px.
- Icons in popups and prompts: 60px x 60px.
- Dial phone number icon: 12px x 36px.



Generally, images should be optimized for OpenStage 80 phones. On OpenStage 60, they will be resized correctly.

5.23 Player

The player is designed for applications that involve audio streaming. All player commands issued by the user are sent to the server, enabling the server to control the streaming accordingly.

The player is a transient screen, replacing the current screen on the display. However, those parts of the current screen which are not covered by the player are still visible in the background. The player displays the current progress through a media stream, and offers the functions:

- play,
- pause,
- stop,
- rewind,
- skip rewind,
- forward,
- skip forward,
- call,
- delete,

similar to common audio players.

The user interface consists of a gauge and a number of buttons. The gauge may be interactive or non-interactive. The non-interactive gauge value is automatically updated on the phone, provided a time-based maximum value is supplied. For further information, please refer to Section 5.21, "Form/Gauge".

The player has eight modes of operation, which determine the state, behaviour, and appearance of the player:

- playing,
- edit,
- stopped playing,
- recording,
- stopped recording,
- playback recording,
- edit playback,

- disabled.

The disabled mode is controlled locally on the phone, whereas the other seven modes are controlled by the remote program.

Three states are defined for the player:

- PLAYING,
- RECORDING,
- STOPPED.

Additionally, two modes are defined by the `Mode` attribute:

- CALL
- RECORD

The player buttons and the TouchGuide allow user interaction with the server-side program. Whenever the user presses a TouchGuide or player button, a specific key/value pair, consisting of the player's `Key` attribute and the function type, is immediately sent to the specified URL. If, for instance, the value of the `Key` attribute is `MyPlayer`, and the user presses the forward button, the key/value pair is `MyPlayer=forward`. Also the key/value pairs of the gauge (see Section 5.21, "Form/Gauge") and of any hiddens (see Section 5.11, "Hidden") are sent to the URL.

XML Object Reference

Player

The following table shows the buttons and the related function types, i. e. the value of the key/value pair whose key is the value of `Key`:

Player Button	Function
	delete
	rewind
	skip_rewind
	play
	pause
	stop
	skip_forward
	forward
	call
	confirm

Table 5-5





Player Button	Function
	record

Table 5-5

The following table shows the TouchGuide buttons and the related function types, i. e. the value of the key/value pair whose key is the value of `Key`:

TouchGuide Button	Function
	next
	previous
	cancel

The following attributes are available to the `IppPlayer` element:

Attribute	Value	Function/Remarks
Mode (mandatory)	CALL	With this value, the phone can be in playing, edit, and stopped mode.
	RECORD	With this value, the phone can be in recording, stopped record, edit playback, and playback recording mode.
State (mandatory)	PLAYING	All buttons are displayed except for the play button, and the gauge is automatically updating.
	STOPPED	All buttons are displayed except for the stop button, and the gauge is not automatically updating.
	RECORDING	Only the stop and the delete button are displayed, and the gauge is automatically updating.

Attribute	Value	Function/Remarks
Key (mandatory)	Key for the Player	Together with the function type of the player button that has been pressed, the key forms a key/value pair. Please note that the key must be unique within a screen.

The following table shows the possible player modes and their corresponding interactive, state, and mode values:

Player Mode	Attributes		
	Gauge - Interactive	State	Mode
Playing	AUTO	PLAYING	CALL
Edit	USER	STOPPED	CALL
Stopped	AUTO	STOPPED	CALL
Recording	AUTO	RECORDING	RECORD
Stopped Record	AUTO	STOPPED	RECORD
Edit Playback	USER	STOPPED	RECORD
Playback Re-cording	AUTO	PLAYING	RECORD

Table 5-6

Syntax

```
<IppPlayer State="PLAYING | STOPPED | RECORDING" Mode="CALL | RECORD"
Key=" [Key] ">
  <Url> </Url>
  <IppGauge> </IppGauge>
</IppPlayer>
```

Child Elements

- Url

The URL invoked when a button is pressed.

- IppGauge

The `Interactive` attribute must be set to `USER` or `AUTO`, according to the current state and mode of the Player. For details, please refer to Table 5-5.

Example

```

<IppPhone>
  <IppDisplay InitialScreen="1">
    <IppScreen ID="1" HiddenCount="0" CommandCount="2">
      <IppForm ItemCount="2">
        <Title>Play / Record</Title>
        <Url>http://subdomain.domain/path/program</Url>
        <IppItem CommandCount="1">
          <IppStringItem>
            <Label></Label>
            <Text>Play</Text>
          </IppStringItem>
          <IppCommand Type="SCREEN" DisplayOn="LISTITEM"
Priority="0">
            <Label>Select</Label>
            <ScreenID>2</ScreenID>
          </IppCommand>
        </IppItem>
        <IppItem CommandCount="1">
          <IppStringItem>
            <Label></Label>
            <Text>Record</Text>
          </IppStringItem>
          <IppCommand Type="SCREEN" DisplayOn="LISTITEM" Pri-
ority="0">
            <Label>Select</Label>
            <ScreenID>3</ScreenID>
          </IppCommand>
        </IppItem>
      </IppForm>
    </IppScreen>
    <IppScreen ID="2" HiddenCount="1" CommandCount="2">
      <IppPlayer State="PLAYING" Mode="CALL" Key="Myplayer">
        <Url>http://subdomain.domain/path/program</Url>
        <IppGauge Interactive="AUTO" Key="G1">
          <Label></Label>
          <Maximum>100</Maximum>
          <Initial>0</Initial>
        </IppGauge>
      </IppPlayer>
    </IppScreen>
  </IppDisplay>
</IppPhone>

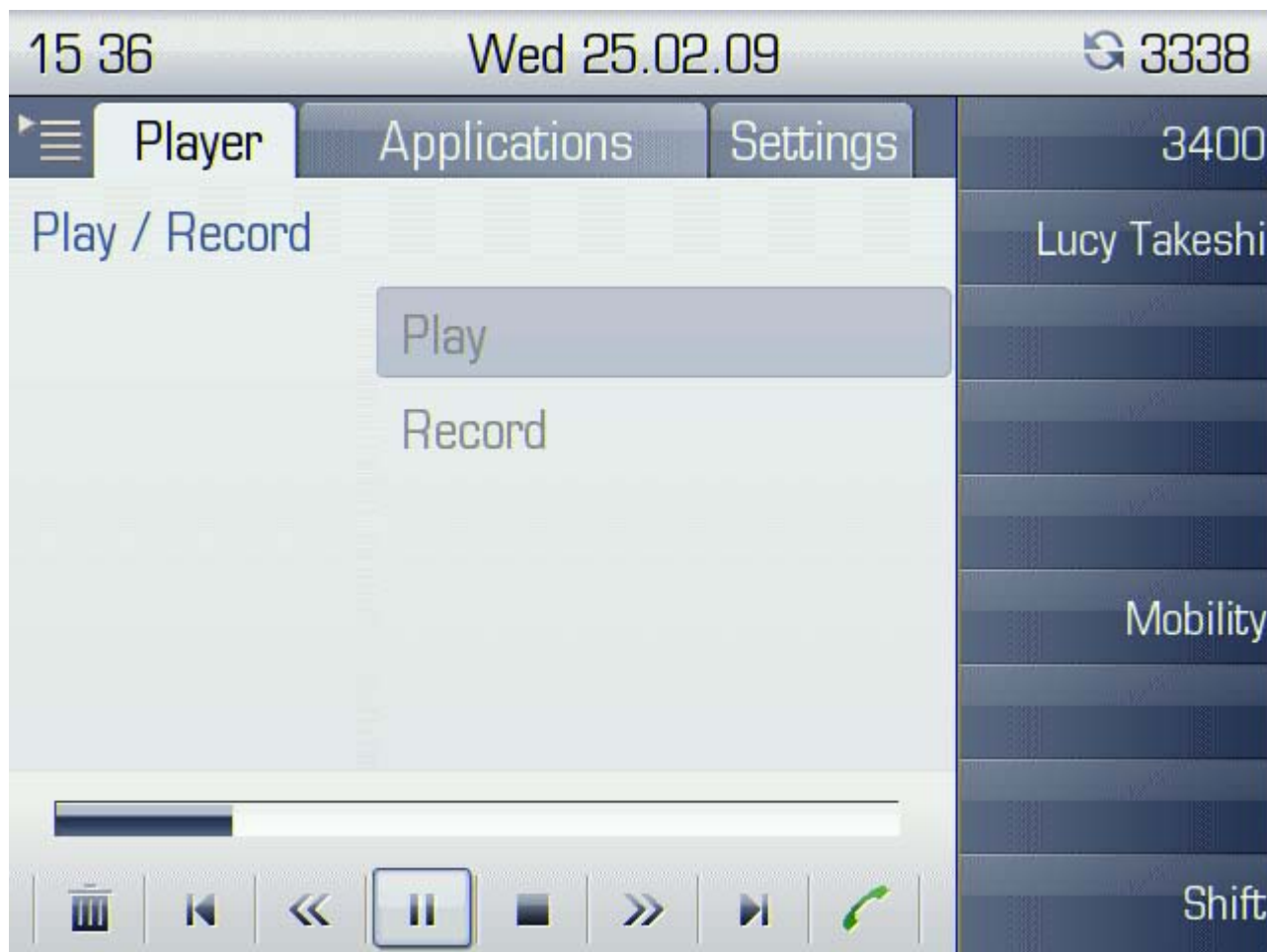
```


XML Object Reference

Player

```
<IppScreen ID="3" HiddenCount="1" CommandCount="2">
  <IppPlayer State="RECORDING" Mode="RECORD" Key="Myplayer">
    <Url>http://subdomain.domain/path/program</Url>
    <IppGauge Interactive="AUTO" Key="G1">
      <Label>Interactive Gauge</Label>
      <Maximum>100</Maximum>
      <Initial>0</Initial>
    </IppGauge>
  </IppPlayer>
</IppScreen>
</IppDisplay>
</IppPhone>
```

Screenshot



5.24 Phone Number

This element enables the translation of a given phone number which is stored in the local phone book. The phone number can be translated to:

- the name of the participant,
- the picture clip associated to the participant, if available,
- or, alternatively, an icon of the phone type (e.g. mobile number, home number, etc.).

If the translation fails, alternative text can be displayed instead of the phone number.

Depending on the GUI element wherein the phone number is placed, some or all of the information listed above may be displayed instead of or in addition to the phone number.

Examples:

- An alert can display the name, number, and picture clip of the participant (if available), or a phone type icon (see Section 5.7, "Alert").



If the name and number are both displayed, the name is placed on the top line, and the number is placed immediately below it.

- A single column list or the left hand column in a multiple column list can display a picture or an icon followed by the name or number (see Section 5.8, "List").
- Any other column in a multiple column list can only display the name or number (see Section 5.8, "List").

Attributes

The following attributes are available to the `IppPhoneNumber` element:

Attribute	Value	Function/Remarks
ImageType (optional)	PHONETYPE	The icon of the phone type according to the given phone number is displayed.
	PICTURECLIP	The picture clip associated to the participant is displayed, if available.
NumberType (optional)	NUMBER	Only the number is displayed.
	NAME	Only the corresponding name in the phone book is displayed.
	BOTH	Both the number and the corresponding name in the phone book are displayed



For consistency with the style for presentations of call identities on OpenStage, `ImageType` should equal `PICTURECLIP` and `NumberType` should equal `NAME`.

Syntax

```
<IppPhoneNumber ImageType=" [ PHONETYPE | PICTURECLIP ] "  
NumberType=" [ NUMBER | NAME | BOTH ] ">  
  <AltText> </AltText>  
</IppPhoneNumber>
```

Child elements

- `AltText`

In case the number does not exist in the phone book, the alternative text is displayed instead of the number, name or picture clip.

6 Push Capability

An XML application can be started or updated by a remote program, without any user action. For this purpose, the server-side program sends a specific HTTP or HTTPS message to the corresponding web service on the phone. For detailed information on the composition of a push request, see Section 6.2, "The Push Request".

6.1 Push Types

There are four different types of push requests. The type determines under which circumstances the information is displayed to the user.

The four different push types and their characteristics are:

1. **Active:** If the required XML application is not running, nothing changes; the application is not started by the push request. The information is displayed in a non-intrusive manner; the application will not gain the focus on push.

This type is appropriate for non-essential information which should only be displayed if the required XML application is already running.

2. **Queue:** If the required XML application is not running, it is started by the push request. However, the application will not gain the focus on push. Alert popups (see Section 5.7, "Alert") and players (see Section 5.23, "Player") are not opened on push.

This type is appropriate for information that does not have to be presented immediately to the user (e.g. offers, restaurant menu, message of the day).

3. **Indicate:** If the required XML application is not running, it is started by the push request. However, the application will not gain the focus on push. The information is displayed to the user in a non-intrusive manner.

This type is appropriate for non-essential information (e.g. new E-mail received, message playback progress).

4. **Force:** If the required XML application is not running, it is started by the push request. Additionally, the application will gain the focus on push.

This type is appropriate for urgent information that must be immediately displayed to the user.

The following table shows which actions are performed by which push type, considering the possible sub-conditions:

Push Capability

Push Types

Action	sub-condition		'Push' Type		
		Active	Queue	Indicate	Force
XML application started on push	Application is configured and not running	No	Yes	Yes	Yes
	Application is not configured	Error case - the push request is ignored.			
XML document loaded from the nominated URL on the server		Yes, if program is already running.	Yes	Yes	Yes

Table 6-1

Action	sub-condition	'Push' Type			
		Active	Queue	Indicate	Force
XML document rendered	XML document contains list (see Chapter 5.8), text box (see Chapter 5.9), or form (see Chapter 5.12)	Yes, application tab updated if application is already running.	Yes, application tab updated.	Yes, application tab updated.	Yes, application tab updated.
	XML document contains alert (see Chapter 5.7)	Popup is only created if program is already running. Displayed only if no other popup or prompt is showing.	No popup is created. The previously displayed screen is shown, if applicable.	Popup is created. Displayed only if no other popup or prompt is showing.	Popup is created. Displayed only if no other popup or prompt is showing.
	XML document contains player (see Chapter 5.23)	Player is created if program is already running. Displayed only if no other popup or prompt is showing.	No player is displayed. The previously displayed screen is shown.	Player is created. Displayed only if no other popup or prompt is showing.	Player is created. Displayed only if no other popup or prompt is showing.
XML application gains screen focus		No	No	No	Yes

Table 6-1

Push Capability

Push Types

Action	sub-condition	'Push' Type			
		Active	Queue	Indicate	Force
Screensaver is running		Stopped for alert popup or player if the application is already running, with the following constraint: When the push request starts a player, the screensaver will only stop if the XML application has had the focus before the screensaver had started.	Stopped if the push request has started the application. Neither alert nor popup player are started by push requests.	Stopped if the push request has started the application, and for alert popup or player, with the following constraint: When the push request starts a player, the screensaver will only stop if the XML application has had the focus before the screensaver had started.	Stopped
Phone locked		No change	No change	No change	No change

Table 6-1

6.2 The Push Request

The push request is an HTTP or HTTPS POST request sent to the following phone-side URL:

`http://<ip-address>:<port-number>/server_push.html/ServerPush`

Example:

`http://137.223.234.11:8085/server_push.html/ServerPush`

The Push request must contain the following Information:

Key	Value	Function
ServerAddr	IP address	IP address of the server where the remote program is located.
ServerPort	Port number	Port number of the server where the remote program is located.
ProgramName	Resource name	Entity part of the URI pointing to the server-side program. For instance, this can be an XML file name, or the name of a servlet. Additional parameters can be added like this: <code>servlet?id=117&param1=hi</code> If a tab on demand is to be started, this parameter must contain the appropriate application name (Tab 1...3 Application name , see Section 3.1, "Configuring an Application on the Phone" and Section 3.4, "Application Suite Using Tabs On Demand").

Table 6-2

Push Capability

The Push Request

Key	Value	Function
RequestType	active	The XML document is loaded, if the program is already started. If the document contains a popup (IppAlert) or a player (IppPlayer), and no other popup or player is shown already, the player or popup is displayed. For detailed information, see the table in Section 6.1, "Active".
	queue	The program is started in the background and the XML document is loaded. For detailed information, see the table in Section 6.1, "Queue".
	indicate	The program is started and the XML document is loaded. If the document contains a popup (IppAlert) or a player (IppPlayer), and no other popup or player is shown already, the player or popup is displayed. For detailed information, see the table in Section 6.1, "Indicate".
	force	The program is started and the XML document is loaded. It is displayed to the user with input focus, unless a prompt (e.g. incoming call Alert or Dialer) is showing. For detailed information, see the table in Section 6.1, "Force".
MidletName	Name of the MIDlet	Name of the application that is to be controlled by the Push request. Depending on the type of Push request, and the current state of the application, the application may be started by the Push request. This parameter is equivalent to the Application name entered when the application was configured on the telephone (see Section 3.1, "Configuring an Application on the Phone").
ServerProtocol	Communication protocol	Indicates the protocol used by the server for communicating to the phone. Examples: http, https (future).
ServerContextKey	Key	The key for the key/value pair that the phone will send to the server-side program. The program can use this key/value pair to detect whether the XML document request goes back to this push request.
ServerContextValue	Value	The value for the key/value pair that the phone will send to the server-side program.

Table 6-2

Push Request Example:

```
POST /server_push.html/ServerPush HTTP/1.1
User-Agent: Java/1.6.0_20
Host: 192.168.1.244:8085
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-type: application/x-www-form-urlencoded
Content-Length: 204
MidletName=testxml&ServerProtocol=http&
ServerAddr=192.168.1.151&ServerPort=8080&
ProgramName=testxml/servlet&RequestType=force&
ServerContextKey=usr;key&
ServerContextValue=RFT;0002hagdtefsjuhsjdtfgrhywhdbzu
```


7 Using Additional Phone Keys




7.1 Overview

In addition to the input and navigation elements accessible to OpenStage XML applications, the free programmable function keys (FPK) and, with firmware V2R1 onwards, the fixed function keys (FFK) can be utilized for communication with a server-side program. For this purpose, a function that sends a configurable HTTP or HTTPS request to a remote server is assigned to the key.

To enable feedback from the server, the LED associated with the key can be controlled remotely. This can be done via SIP notify messages, or, with firmware version V2R2 or higher, via a combination of HTTP push requests (see Section 7.4, "Creating the Push Request for LED Control (V2R2)") and XML documents (see Section 7.5, "XML Document for LED Control").

7.2 Configuring the Phone

To assign this function to a free programmable key, open the web based management (see Section 3.1.3, "Using the Web Based Management (WBM)", step 1-3) and navigate to **System > Features > Program keys**. In the menu for the desired key (normal or shifted level, as appropriate), select the function **Send URL**.

To assign the function to a fixed function key, navigate to **System > Features > Fixed Keys**. The forwarding key , the release key , and the voice recognition key  can be configured here. In the menu for the desired key, select the function **Send URL**.

The following parameters are configurable for the **Send URL** function:

- In **Key label <n>**, the label for this key is defined.
- **Protocol** defines whether HTTP or HTTPS will be used for sending the URL to the server.
- The **Web server address** is the IP address or DNS name of the remote server to which the URL is to be sent.
- The **Port** is the target port at the server to which the URL is to be sent.
- The **Path** is the server-side path to the desired function, i. e. the part of the URL that follows the IP address or DNS name. Example: `webpage/checkin`
With firmware version V2R2, this is the path that returns the XML document containing the LED status information.


- In the **Parameters** field, one or more key/value pairs in the format `<key>=<value>` can be added to the request, separated by an ampersand (&).
Example: `mode=remote&action=start`



The question mark will be automatically added between the path and the parameters. If a question mark has been entered at the start of the parameters, it will be stripped off automatically.

- **Method** determines the HTTP method to be used, which can either be GET or POST. If GET is selected, the additional parameters (**Parameters**) and the user id/password (**Web server user ID/Web server password**) are part of the URL. If POST is selected, these data form the body of the message.
- In case the web server requires user authentication, the parameters **Web server user ID** and **Web server password** can be used. If not null, the values are appended between the server-side path (**Path**) and the additional parameters (**Parameter**).
- If the **LED controller URI** is given, the LED associated with this key indicates the state of the call number or SIP URI specified here, provided the SIP server sends a notification (SIP NOTIFY, based on RFC 4235):
 - Busy notification (state=confirmed): LED is glowing.
 - Ringing notification (state=early): LED is blinking.
 - Idle notification (state=terminated): LED is dark.



When assigning the function described here to the release key , please consider that this key has no LED.

- With firmware version V2R2, the **Push support** parameter is available. If activated, the LED is controllable by a combination of an HTTP push request and an XML document (for details, see Section 7.4, "Creating the Push Request for LED Control (V2R2)" and Section 7.5, "XML Document for LED Control").



If you want to use the HTTP push solution, please ensure that the **LED controller URI** field is empty. Otherwise, the phone will only use the SIP mechanism for LED control, and ignore the push request.

- The **Symbolic name**, which is available with firmware version V2R2, is used to assign a push request from the application server to the appropriate free programmable key resp. fixed function key. This value must be unique for all keys involved.

7.3 Phone Request

OpenStage phones can send GET or POST requests, depending on the configuration. The specific request string is sent in the query string when GET is chosen, and in the message body when POST is chosen. The `Content-type` of the POST request is `text/xml`.

With firmware version V2R1, the phone sends an HTTP(S) request when the user presses the key to which the **Send URL** function is assigned. The request string contains the parameters **Web server user ID**, **Web server password**, and **Parameters**; for example:

```
userid=jdoe&password=00secret&mode=remote&action=start
```

With firmware version V2R2 onwards, the phone sends an HTTP(S) request in any of the following cases:

- The user presses the key to which the **Send URL** function is assigned. The request string contains the **Web server user ID**, the **Web server password**, the **Parameters**, the phone's IP address, the phone's call number, and the **Symbolic Name**; for example:

```
userid=jdoe&password=00secret&mode=remote&action=start&  
ipaddress=192.168.1.244&phonenumber=3338&symbn=key4
```

- The server-side program has sent a push request to the phone. The request string is the same as in the case of key press.
- The LED state must be updated or restored due to a configuration change or restart. The phone indicates this to the server by adding `updt` to the request string. Thus, the request string contains the **Web server user ID**, the **Web server password**, the **Parameters**, the phone's IP address, the phone's call number, the **Symbolic Name**, and the `updt` flag; for example:

```
userid=jdoe&password=00secret&mode=remote&action=start&  
ipaddress=192.168.1.244&phonenumber=3338&symbn=key4&updt
```


7.4 Creating the Push Request for LED Control (V2R2)

The state of the LED is stored and controlled by a remote XML document. The phone will request this document if the user presses a **Send URL** key, or if it has received an appropriate push request. This is a special HTTP POST message, similar to the push request for XML applications. It must contain the following key/value pairs:

- `ServerAddr`: The IP address or DNS name of the server that provides the XML document for LED control. At the phone's web interface, this parameter is configured in **Program keys > Send URL > Web server address**.
- `ServerPort`: The port at which the server program is listening. At the phone's web interface, this parameter is configured in **Program keys > Send URL > Port**.
- `ProgramName`: A symbolic name used to assign a push request from the application server to the appropriate key. At the phone's web interface, this parameter is configured in **Program keys > Send URL > Symbolic name**.
- `MidletName`: A symbolic name used to assign a push request from the application server to the appropriate key; the value is identical with that of `ProgramName`. At the phone's web interface, this parameter is configured in **Program keys > Send URL > Symbolic name**.
- `RequestType`: Must be set to `sendURL`.

Example:

```
MidletName=key4&ServerProtocol=http&ServerAddr=192.168.1.151&  
ServerPort=80&ProgramName=key4&RequestType=sendURL
```


7.5 XML Document for LED Control

The desired state for the LED assigned to the key is stored in a XML document which is to be delivered from the server to the phone when it has sent an appropriate HTTP(S) request. The design of this XML structure is due to the possibility of future enhancements.

Syntax Example

```
<Batch>
  <Capabilities_Req TYPE="ANY_OF">
    <Capability NAME="PHONE_TYPE_OS80" />
    <Capability NAME="PHONE_TYPE_OS60" />
  </Capabilities_Req>
  <Capabilities_Req TYPE="ALL_OF">
    <Capability NAME="XML_LED_STATE_CONTROL" />
  </Capabilities_Req>
  <Remote_Control>
    <Turnled ID="led04" STATE="ON" />
  </Remote_Control>
</Batch>
```

"Batch" Element

Serves as root element.

"Capabilities_Req" Element

Optional element which contains a list of capabilities required by the web application. If omitted, the web application assumes that all phones can handle the tasks in this document. The element has a `Type` attribute which specifies in what way the capabilities are required.

Attribute	Value	Function/Remarks
Type	ANY_OF	The phone must have at least one capability or feature from the list.
	ALL_OF	The phone must have all capabilities or features from the list.

"Capability" Element

This object describes one particular capability. Each capability must have the `NAME` attribute, which specifies a capability or the phone type. The following attributes are possible:

Attribute	Value	Function/Remarks
NAME	PHONE_TYPE_OS80	The phone is an OpenStage 80.
	PHONE_TYPE_OS60	The phone is an OpenStage 60.
	PHONE_TYPE_OS40	The phone is an OpenStage 40.
	PHONE_TYPE_OS20	The phone is an OpenStage 20.
	PHONE_TYPE_OS15	The phone is an OpenStage 15.
	PHONE_TYPE_OS10	The phone is an OpenStage 10.
	XML_LED_STATE_CONTROL	The phone has the capability to change the state of LEDs associated with FPKs or FFKs.

"Remote_Control" Element

Contains a list of tasks which are destined to change the state of the phone. Currently, only `Turnled` is supported.

"Turnled" Element

This element controls the LED status. The following attributes are possible:

Attribute	Value	Function/Remarks
ID	String	An ID can be assigned to the LED. Currently, this attribute is not used; however, it is required by the XML schema (see Section 8.3, "XML Schema for the Send URL Feature").
STATE	ON	LED is glowing.
	OFF	LED is dark.
	FLASH	LED is blinking.

8 Appendix

8.1 Glossary

Term/Abbreviation	Definition/Description
FFK	
FPK	
HFA	HiPath Feature Access
jad	Java Application Descriptor
Jar	Java Archive File
SIP	Session Initiation Protocol
Web Based Management	Built-in configuration and administration facility which is accessible by a web browser.

8.2 XML Schema for XML Applications

This appendix contains the entire XML schema supported by the HFA and SIP variants of the OpenStage 60/80.

The XML schema is available for download. For downloading,

1. register with the Community Portal under
[http://www.siemens-enterprise.com/developerportal/Developer-Community/
Register.aspx](http://www.siemens-enterprise.com/developerportal/Developer-Community/Register.aspx)
2. and navigate to
<https://app-enterprise.siemens-enterprise.com/gdmsproxy/retrieve?id=40766600>

Appendix

XML Schema for XML Applications

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      XML Schema for Siemens IP Phone Generic MIDlet. Version 5.0
      Copyright 2007 Siemens Enterprise Communications Limited. All
      rights reserved.
    </xs:documentation>
  </xs:annotation>
  <xs:element name="IppPhone">
    <xs:complexType>
      <xs:sequence>
        <xs:choice>
          <xs:element ref="IppDisplay"/>
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="IppDisplay">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="IppScreen" maxOccurs="5"/>
      </xs:sequence>
      <xs:attribute name="InitialScreen" type="xs:nonNegativeInteger" use="optional"/>
      <xs:attribute name="UpdateScreen" type="xs:nonNegativeInteger" use="optional"/>
    </xs:complexType>
  </xs:element>
```



```
<xs:element name="IppScreen">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="IppKey" minOccurs="0"/>
      <xs:choice>
        <xs:element ref="IppAlert" minOccurs="0"/>
        <xs:element ref="IppList" minOccurs="0"/>
        <xs:element ref="IppTextBox" minOccurs="0"/>
        <xs:element ref="IppForm" minOccurs="0"/>
        <xs:element ref="IppPlayer" minOccurs="0"/>
      </xs:choice>
      <xs:element ref="IppAction" minOccurs="0"/>
      <xs:element ref="IppCommand" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="IppHidden" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="IppTicker" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="HiddenCount" type="xs:nonNegativeInteger" use="required"/>
    <xs:attribute name="CommandCount" type="xs:nonNegativeInteger" use="required"/>
    <xs:attribute name="ID" type="xs:nonNegativeInteger" use="optional" default="0"/>
    <xs:attribute name="Sound" type="xs:string" use="optional"/>
  >
  <!-- default="0"/> -->
  <!-- default="0"/> -->
</xs:complexType>
</xs:element>
```


Appendix

XML Schema for XML Applications

```
<xs:element name="IppAction">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Number" minOccurs="0">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:pattern value="\d*" />
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="Type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="MAKECALL" />
          <xs:enumeration value="ENDCALL" />
          <xs:enumeration value="TURNLEDON" />
          <xs:enumeration value="TURNLEDOFF" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```



```
<xs:element name="IppAlert">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Title" minOccurs="0"/>
      <xs:element ref="Url" minOccurs="0"/>
      <xs:element ref="Text" minOccurs="0"/>
      <xs:element ref="Image" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Type" use="optional" default="ERROR">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="ALARM"/>
          <xs:enumeration value="CONFIRMATION"/>
          <xs:enumeration value="ERROR"/>
          <xs:enumeration value="INFO"/>
          <xs:enumeration value="WARNING"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Delay" use="optional" default="FOREVER">
      <xs:simpleType>
        <xs:union memberTypes="xs:positiveInteger">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="FOREVER"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```


Appendix

XML Schema for XML Applications

```
<xs:element name="IppButton">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Label" minOccurs="0"/>
      <xs:element ref="Image" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Layout" use="optional" default="DE-
FAULT">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="CENTER"/>
          <xs:enumeration value="DEFAULT"/>
          <xs:enumeration value="LEFT"/>
          <xs:enumeration value="NEWLINE_AFTER"/>
          <xs:enumeration value="NEWLINE_BEFORE"/>
          <xs:enumeration value="RIGHT"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Type" use="optional" default="IMAGE">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="IMAGE"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref="Key" use="required"/>
    <xs:attribute ref="Value" use="required"/>
  </xs:complexType>
</xs:element>
```



```
<xs:element name="IppChoiceGroup">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Label" minOccurs="0"/>
      <xs:element ref="Option" minOccurs="0" maxOccurs="un-
bounded"/>
    </xs:sequence>
    <xs:attribute name="Numbered" default="NO">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="NO"/>
          <xs:enumeration value="YES"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Type" use="required">
      <!-- default="EXCLUSIVE"> -->
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="EXCLUSIVE"/>
          <xs:enumeration value="MULTIPLE"/>
          <xs:enumeration value="POPUP"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Count" type="xs:nonNegativeInteger"
use="required"/>
      <!-- default="0"/> -->
    </xs:complexType>
  </xs:element>
```


Appendix

XML Schema for XML Applications

```
<xs:element name="IppCommand">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Label" type="xs:string" minOccurs="0"/>
      <xs:element name="ScreenID" type="xs:nonNegativeInteger"
minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="SELECT"/>
          <xs:enumeration value="BACK"/>
          <xs:enumeration value="UPDATE"/>
          <xs:enumeration value="SCREEN"/>
          <xs:enumeration value="CANCEL"/>
          <xs:enumeration value="EXIT"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Priority" use="optional" default="0">
      <xs:simpleType>
        <xs:restriction base="xs:nonNegativeInteger">
          <xs:maxInclusive value="9"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Auto" use="optional" default="0">
      <xs:simpleType>
        <xs:restriction base="xs:nonNegativeInteger"/>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="DisplayOn" use="optional" default="OP-
TIONS">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="OPTIONS"/>
          <xs:enumeration value="LISTITEM"/>
          <xs:enumeration value="BOTH"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Select" use="optional" default="NO">
      <xs:simpleType>
        <xs:restriction base="xs:string">
```



```
        <xs:enumeration value="YES"/>
        <xs:enumeration value="NO"/>
    </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="Default" use="optional" default="NO">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="YES"/>
            <xs:enumeration value="NO"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<xs:attribute ref="Key" use="optional"/>
<xs:attribute ref="Value" use="optional"/>
</xs:complexType>
</xs:element>
```


Appendix

XML Schema for XML Applications

```
<xs:element name="IppDateField">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Label" minOccurs="0"/>
      <xs:element ref="Timezone" minOccurs="0"/>
      <xs:element ref="Date" minOccurs="0"/>
      <xs:element ref="Time" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Mode" use="optional" default="DATETIME">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="DATE"/>
          <xs:enumeration value="TIME"/>
          <xs:enumeration value="DATETIME"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Default" use="optional" default="NULL">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="NULL"/>
          <xs:enumeration value="MODE"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref="DateKey" use="optional"/>
    <xs:attribute ref="TimeKey" use="optional"/>
  </xs:complexType>
</xs:element>
```



```
<xs:element name="IppForm">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Title" minOccurs="0"/>
      <xs:element ref="Url" minOccurs="0"/>
      <!-- IppForm may contain any combination of none, one, or
      more IppButton, IppChoiceGroup, IppDateField, IppGauge,
      IppImageItem, IppItem, IppSpacer, IppStringItem, and
      Ipp TextField in any order. The correct XML syntax should
      surround each one by "<IppItem></IppItem>". However, this
      increases the size of the XML document and has an impact
      on performance.
      When validating a simple IppForm with only one type of
      element, each element does not have to be surrounded by
      <IppItem></IppItem>. When validating a complex IppForm
      with many different types of elements, each elements must
      be surrounded by <IppItem></IppItem>. When the XML
      document for a complex IppForm has been validated, the
      "IppItem" tags can be removed from the XML document. If
      they remain in the XML document, they will be correctly
      parsed.-->
      <xs:choice>
        <xs:element ref="IppButton" minOccurs="0" maxOc-
curs="unbounded"/>
        <xs:element ref="IppChoiceGroup" minOccurs="0" maxOc-
curs="unbounded"/>
        <xs:element ref="IppDateField" minOccurs="0" maxOc-
curs="unbounded"/>
        <xs:element ref="IppGauge" minOccurs="0" maxOc-
curs="unbounded"/>
        <xs:element ref="IppImageItem" minOccurs="0" maxOc-
curs="unbounded"/>
        <xs:element ref="IppItem" minOccurs="0" maxOccurs="un-
bounded"/>
        <xs:element ref="IppSpacer" minOccurs="0" maxOc-
curs="unbounded"/>
        <xs:element ref="IppStringItem" minOccurs="0" maxOc-
curs="unbounded"/>
        <xs:element ref="IppTextField" minOccurs="0" maxOc-
curs="unbounded"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="ItemCount" type="xs:nonNegativeInteger"
use="required"/>

```


Appendix

XML Schema for XML Applications

```

    <xs:attribute name="Proportion" use="optional" de-
fault="40_60">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="40_60"/>
            <xs:enumeration value="50_50"/>
            <xs:enumeration value="60_40"/>
            <xs:enumeration value="75_25"/>
            <xs:enumeration value="25_75"/>
            <xs:enumeration value="15_85"/>
            <xs:enumeration value="0_100"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<!-- default="0"/> -->
</xs:complexType>
</xs:element>

```



```
<xs:element name="IppGauge">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Label" minOccurs="0"/>
      <xs:element ref="Maximum" minOccurs="0"/>
      <xs:element ref="Initial" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Interactive" use="optional" default="US-
ER">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="USER"/>
          <xs:enumeration value="AUTO"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref="Key" use="optional"/>
  </xs:complexType>
</xs:element>

<xs:element name="IppHidden">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Value" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Type" use="required">
      <!-- default="VALUE"> -->
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="VALUE"/>
          <xs:enumeration value="PHONENUMBER"/>
          <xs:enumeration value="IPADDRESS"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref="Key" use="required"/>
  </xs:complexType>
</xs:element>
```


Appendix

XML Schema for XML Applications

```
<xs:element name="IppImageItem">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Label" minOccurs="0"/>
      <xs:element ref="Image" minOccurs="0"/>
      <xs:element ref="AltText" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Layout" use="optional" default="CENTER">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="CENTER"/>
          <xs:enumeration value="DEFAULT"/>
          <xs:enumeration value="LEFT"/>
          <xs:enumeration value="NEWLINE_AFTER"/>
          <xs:enumeration value="NEWLINE_BEFORE"/>
          <xs:enumeration value="RIGHT"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<xs:element name="IppItem">
  <xs:complexType>
    <xs:sequence>
      <xs:choice>
        <xs:element ref="IppButton" minOccurs="0"/>
        <xs:element ref="IppChoiceGroup" minOccurs="0"/>
        <xs:element ref="IppDateField" minOccurs="0"/>
        <xs:element ref="IppGauge" minOccurs="0"/>
        <xs:element ref="IppImageItem" minOccurs="0"/>
        <xs:element ref="IppSpacer" minOccurs="0"/>
        <xs:element ref="IppStringItem" minOccurs="0"/>
        <xs:element ref="IppTextField" minOccurs="0"/>
      </xs:choice>
      <xs:element ref="IppCommand" minOccurs="0" maxOccurs="un-
bounded"/>
    </xs:sequence>
    <xs:attribute name="CommandCount" type="xs:nonNegativeInte-
ger" use="required"/>
    <!-- default="0"/> -->
  </xs:complexType>
</xs:element>
```



```
<xs:element name="IppKey">
  <xs:complexType>
    <xs:attribute name="Keypad" use="required">
      <!--default="NO"> -->
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="NO"/>
          <xs:enumeration value="YES"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="SendKeys" use="required">
      <!--default="NO"> -->
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="NO"/>
          <xs:enumeration value="YES"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="BufferKeys" use="optional" default="NO">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="NO"/>
          <xs:enumeration value="YES"/>
          <xs:enumeration value="SUBSEQUENT"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="BufferLength" use="optional" de-
fault="0">
      <xs:simpleType>
        <xs:restriction base="xs:nonNegativeInteger"/>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="TermKey" use="optional" default=" ">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="[ *#0-9]{1}"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```


Appendix

XML Schema for XML Applications

```
<xs:attribute ref="UrlKey" use="required"/>
</xs:complexType>
</xs:element>

<xs:element name="IppList">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Title" minOccurs="0"/>
      <xs:element ref="Url" minOccurs="0"/>
      <xs:element ref="Option" minOccurs="0" maxOccurs="un-
bounded"/>
    </xs:sequence>
    <xs:attribute name="Numbered" use="optional" default="NO">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="NO"/>
          <xs:enumeration value="YES"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Type" use="required">
      <!-- default="IMPLICIT"> -->
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="IMPLICIT"/>
          <xs:enumeration value="EXCLUSIVE"/>
          <xs:enumeration value="MULTIPLE"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Count" type="xs:nonNegativeInteger"
use="required"/>
    <xs:attribute name="Columns" use="optional" default="1">
      <xs:simpleType>
        <xs:restriction base="xs:nonNegativeInteger">
          <xs:enumeration value="1"/>
          <xs:enumeration value="3"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref="FColor"/>
    <xs:attribute ref="TColor"/>
    <xs:attribute ref="SFCColor"/>
    <xs:attribute ref="STColor"/>
  </xs:complexType>
</xs:element>
```



```
        <!-- default="0"/> -->
    </xs:complexType>
</xs:element>

<xs:element name="IppPhoneNumber">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="AltText" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="ImageType" use="optional" default="NOIM-
AGE">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="NOIMAGE"/>
                    <xs:enumeration value="PHONETYPE"/>
                    <xs:enumeration value="PICTURECLIP"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="NumberType" use="optional" default="NUM-
BER">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="NUMBER"/>
                    <xs:enumeration value="NAME"/>
                    <xs:enumeration value="BOTH"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
</xs:element>
```


Appendix

XML Schema for XML Applications

```
<xs:element name="IppPlayer">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Url" minOccurs="0"/>
      <xs:element ref="IppGauge" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="State" use="required">
      <!-- default="STOPPED"> -->
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="PLAYING"/>
          <xs:enumeration value="RECORDING"/>
          <xs:enumeration value="STOPPED"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Mode" use="required">
      <!-- default="CALL"> -->
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="CALL"/>
          <xs:enumeration value="RECORD"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref="Key" use="required"/>
  </xs:complexType>
</xs:element>
```



```
<xs:element name="IppSpacer">
  <xs:complexType>
    <xs:attribute name="NewLine" use="required">
      <!-- default="NEWLINE_BEF_AFT" -->
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="NEWLINE_AFTER"/>
          <xs:enumeration value="NEWLINE_BEFORE"/>
          <xs:enumeration value="NEWLINE_BEF_AFT"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Height" use="optional" default="0">
      <xs:simpleType>
        <xs:restriction base="xs:nonNegativeInteger"/>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Width" use="optional" default="0">
      <xs:simpleType>
        <xs:restriction base="xs:nonNegativeInteger"/>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```


Appendix

XML Schema for XML Applications

```
<xs:element name="IppStringItem">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Label" minOccurs="0"/>
      <xs:element ref="Text" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="IppTextBox">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Title" minOccurs="0"/>
      <xs:element ref="Text" minOccurs="0"/>
      <xs:element ref="Url" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="MaxSize" type="xs:nonNegativeInteger"
use="optional"/>
    <xs:attribute name="Constraint" use="optional" de-
fault="ANY">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="ANY"/>
          <xs:enumeration value="NUMERIC"/>
          <xs:enumeration value="PASSWORD"/>
          <xs:enumeration value="PHONENUMBER"/>
          <xs:enumeration value="URL"/>
          <xs:enumeration value="EMAILADDR"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Password" use="optional" default="NO">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="YES"/>
          <xs:enumeration value="NO"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Uneditable" use="optional" default="NO">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="YES"/>
          <xs:enumeration value="NO"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```



```
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref="Default" use="optional" default="NULL"/>
    <xs:attribute ref="Key" use="optional"/>
</xs:complexType>
</xs:element>
```


Appendix

XML Schema for XML Applications

```
<xs:element name="IppTextField">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Label" minOccurs="0"/>
      <xs:element ref="Text" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="MaxSize" type="xs:nonNegativeInteger"
use="optional"/>
    <xs:attribute name="Constraint" use="optional" de-
fault="ANY">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="ANY"/>
          <xs:enumeration value="NUMERIC"/>
          <xs:enumeration value="PASSWORD"/>
          <xs:enumeration value="PHONENUMBER"/>
          <xs:enumeration value="URL"/>
          <xs:enumeration value="EMAILADDR"/>
          <xs:enumeration value="VALIDATE"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Password" use="optional" default="NO">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="YES"/>
          <xs:enumeration value="NO"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Uneditable" use="optional" default="NO">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="YES"/>
          <xs:enumeration value="NO"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref="Default" use="optional" default="NULL"/>
    <xs:attribute ref="Key" use="optional"/>
  </xs:complexType>
</xs:element>
```



```
<xs:element name="IppTicker">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Text" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="AltText" type="xs:string"/>

<xs:element name="Date">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="\d{4}-\d{2}-\d{2}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="Image">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="imagePath">
        <xs:attribute name="Cache" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="Initial" type="gaugeTimeType"/>
<xs:element name="Label" type="xs:string"/>
<xs:element name="Maximum" type="gaugeTimeType"/>
```


Appendix

XML Schema for XML Applications

```
<xs:element name="Option">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="OptionText" maxOccurs="3"/>
      <xs:element ref="Image" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:nonNegativeInteger"
use="optional"/>
    <xs:attribute name="Selected" use="required">
      <!-- default="FALSE"> -->
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="TRUE"/>
          <xs:enumeration value="FALSE"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref="Key" use="required"/>
    <xs:attribute ref="Value" use="required"/>
    <xs:attribute ref="FColor"/>
    <xs:attribute ref="TColor"/>
    <xs:attribute ref="SFCColor"/>
    <xs:attribute ref="STColor"/>
  </xs:complexType>
</xs:element>
```



```
<xs:element name="OptionText">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element ref="IppPhoneNumber" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="State" default="NORMAL">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="NORMAL"/>
          <xs:enumeration value="GRAY"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<xs:element name="Text">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element ref="IppPhoneNumber" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Time">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="\d{2}:\d{2}:\d{2}.\d{3}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="Timezone" type="timeZoneType"/>
<xs:element name="Title" type="xs:string"/>
<xs:element name="Url" type="xs:anyURI"/>
```


Appendix

XML Schema for XML Applications

```
<xs:element name="Value" type="xs:string"/>
<xs:attribute name="DateKey" type="xs:string"/>
<xs:attribute name="Default">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="NULL"/>
      <xs:enumeration value="TEXT"/>
      <xs:enumeration value="PHONENUMBER"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="FColor" type="colourType"/>
<xs:attribute name="Key" type="xs:string"/>
<xs:attribute name="SFColor" type="colourType"/>
<xs:attribute name="STColor" type="colourType"/>
<xs:attribute name="TColor" type="colourType"/>
<xs:attribute name="TimeKey" type="xs:string"/>
<xs:attribute name="Value" type="xs:string"/>
<xs:attribute name="UrlKey" type="xs:string"/>
```



```
<xs:simpleType name="colourType">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="#[0-9ABCDEF]{6}" />
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="BLACK" />
        <xs:enumeration value="RED" />
        <xs:enumeration value="GREEN" />
        <xs:enumeration value="BROWN" />
        <xs:enumeration value="BLUE" />
        <xs:enumeration value="MAGENTA" />
        <xs:enumeration value="CYAN" />
        <xs:enumeration value="LIGHTGRAY" />
        <xs:enumeration value="DARKGRAY" />
        <xs:enumeration value="LIGHTRED" />
        <xs:enumeration value="LIGHTGREEN" />
        <xs:enumeration value="YELLOW" />
        <xs:enumeration value="LIGHTBLUE" />
        <xs:enumeration value="LIGHTMAGENTA" />
        <xs:enumeration value="LIGHTCYAN" />
        <xs:enumeration value="WHITE" />
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>

<xs:simpleType name="gaugeTimeType">
  <xs:union memberTypes="xs:nonNegativeInteger">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="\d{2}:\d{2}" />
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```


Appendix

XML Schema for XML Applications

```
<xs:simpleType name="imagePath">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="."/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="file:."/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="http:."/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="map:."/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>

<xs:simpleType name="timeZoneType">
  <xs:union memberTypes="xs:string">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="GMT[\\+|\\-]\\d*" />
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
</xs:schema>
```


8.3 XML Schema for the Send URL Feature

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNam-
espace="http://xml.netbeans.org/schema/os" xmlns:tns="http://xml.net-
beans.org/schema/os" elementFormDefault="qualified">

  <xsd:element name="Batch">
    <xsd:complexType>
      <xsd:sequence>

        <xsd:element name="Capabilities_Req" minOccurs="0"
maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>

              <xsd:element name="Capability" minOccurs="0"
maxOccurs="unbounded">
                <xsd:complexType>
                  <xsd:attribute name="NAME" use="required">
                    <xsd:simpleType>
                      <xsd:restriction base="xsd:string">
                        <xsd:enumeration
value="XML_LED_STATE_CONTROL"/>
                        <xsd:enumeration
value="XML_LOGGING"/>
                        <xsd:enumeration
value="XML_SETTINGS"/>
                        <xsd:enumeration
value="PHONE_TYPE_OS80"/>
                        <xsd:enumeration
value="PHONE_TYPE_OS60"/>
                        <xsd:enumeration
value="PHONE_TYPE_OS40"/>
                        <xsd:enumeration
value="PHONE_TYPE_OS20"/>
                        <xsd:enumeration
value="PHONE_TYPE_OS15"/>
                        <xsd:enumeration
value="PHONE_TYPE_OS10"/>
                      </xsd:restriction>
                    </xsd:simpleType>
                  </xsd:attribute>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```


Appendix

XML Schema for the Send URL Feature

```

        </xsd:sequence>
        <xsd:attribute name="TYPE" default="ALL_OF"
use="optional">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="ANY_OF"/>
                    <xsd:enumeration value="ALL_OF"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
</xsd:element>

    <xsd:element name="Remote_Control" minOccurs="0"
maxOccurs="unbounded">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Turnled" minOccurs="0"
maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:attribute name="ID" type="xsd:string"
use="required">
                            </xsd:attribute>
                        <xsd:attribute name="STATE"
use="required">
                            <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                    <xsd:enumeration value="ON"/>
                                    <xsd:enumeration value="OFF"/>
                                    <xsd:enumeration value="FLASH"/>
                                </xsd:restriction>
                            </xsd:simpleType>
                        </xsd:attribute>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

```



```
        <xsd:element name="Settings" minOccurs="0"
maxOccurs="unbounded">
            <xsd:complexType>
                <xsd:sequence>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

    </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```