MGCP Business Phone Packages

Status of this Document

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt
The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

This document describes a collection of MGCP packages that can be used to take advantage of the feature keys and displays on digital business phones and IP-Phones. These packages, when used in conjunction with the packages currently defined in RFC 2705 (Media Gateway Control Protocol Version 1.0), allow an MGCP call-agent to control these types of endpoints.

Table of Contents

1. Introduction
   1.1 General Information
   1.2 Objectives
2. MGCP Packages for Business Phones
   2.1 Feature Key Package
   2.2 Business Phone Package
   2.3 Display XML Package
3. Endpoint Naming
4. Functions that should be Locally Implemented
   4.1 Volume Control
   4.2 Audio Path
5. XML Package Support
   5.1 XML Documents
1. Introduction

The Media Gateway Control Protocol (MGCP) Version 1.0 defines a protocol for controlling Voice over IP Telephony Gateways from external call control elements. As defined, it supports external call control elements called Media Gateway Controllers and assumes that these Gateways can support collections of endpoints. The endpoint type known as an "analog line" can be used as a client interface to provide service to a basic analog telephone unit. The packages that are currently defined to handle events and signals allow only for a basic level of audio connection and signaling to such endpoints. To handle more advanced capabilities commonly found on business phones such as feature keys, speakerphone and displays, it is necessary to define additional packages as extensions to the Version 1.0 MGCP protocol.

The MGCP extension packages defined here are as follows:

- Feature Key Package
o Groups the events and signals associated with the additional keys available on business phones that are non-DTMF and not locally-implemented. These include:
  - Feature Key event allows mapping key numbers to feature keys any phone.
  - Key State signal indicates the state of feature keys.
  - Set Label signal can be used to display a label on the LCD next to a feature key.

- Business Phone Package
  o Groups signals that are not related to feature keys, including:
    - Force Off-hook and Force On-hook signals to allow application integration with speakerphone capabilities.
    - Beep signal to play a beep on the phone.

- Display XML Package
  o Used to convey XML [2] script data to and from the phone to control the display and assign functions to the keys for event reporting.

1.1 General Information

A generic business phone typically includes a number of features that provide access to additional functionality and features useful in the business environment. Beyond the basic handset and dial pad, a Business Phone may optionally include a number of fixed buttons, line keys and programmable feature keys, along with an LCD display and soft-keys.

Specific examples of items that may be included on a Business Phone are:
  - Speakerphone microphone and speaker
  - Speakerphone button and light
  - Message button and light
  - Redial button
  - Volume up and down buttons
  - Hold button and light
  - Transfer button and light
  - Forward button and light
  - Conference button and light
  - Microphone mute button and light
  - Multiple feature keys with lights
  - Multi-line LCD Display
  - Multiple soft-keys next to the LCD display
  - Navigation keys

1.2 Objectives

The high level objectives that were considered in generating the extensions described here are:

- Provide a minimum set of extension packages to the MGCP Version 1.0 protocol to allow applications to take advantage of generic business phone capabilities.

- Provide event and control extensions at a sufficiently low level for an application to implement generic business phone functions without generating excessive or redundant data traffic. (e.g. Sending feature key information on both press and release would be a "don't care" for a call-agent. All it cares about is that
the key was pressed.)

- Provide a mechanism to interface with LCD displays and allow flexibility that will accommodate a variety of application needs and different types of displays available.

2. MGCP Packages for Business Phones

The following packages should be implemented for Business Phones. The G,D,L, and H packages are defined in RFC 2705 [1]. Packages KY, BP and XML are defined in this specification.

<table>
<thead>
<tr>
<th>Package</th>
<th>Name</th>
<th>Defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic Media Package</td>
<td>G</td>
<td>in RFC 2705</td>
</tr>
<tr>
<td>DTMF package</td>
<td>D</td>
<td>in RFC 2705</td>
</tr>
<tr>
<td>Line Package</td>
<td>L</td>
<td>in RFC 2705</td>
</tr>
<tr>
<td>Handset Package</td>
<td>H</td>
<td>in RFC 2705</td>
</tr>
<tr>
<td>Feature Key Package</td>
<td>KY</td>
<td>in this spec</td>
</tr>
<tr>
<td>Business Phone Package</td>
<td>BP</td>
<td>in this spec</td>
</tr>
<tr>
<td>Display XML Package</td>
<td>XML</td>
<td>in this spec</td>
</tr>
</tbody>
</table>

In the tables of events for each package, there are five columns:

- Symbol: the unique symbol used for the event
- Definition: a short description of the event
- R: an x appears in this column is the event can be Requested by the call agent.
- S: if nothing appears in this column for an event, then the event cannot be signaled on command by the call agent. Otherwise, the following symbols identify the type of event:
  - OO On/Off signal. The signal is turned on until commanded by the call agent to turn it off, and vice versa.
  - TO Timeout signal. The signal lasts for a given duration unless it is superseded by a new signal.
  - BR Brief signal. The event has a short, known duration.
- Duration: specifies the duration of TO signals.

2.1 Feature Key Package

Package Name: KY

The Feature Key Package groups the events and signals that are associated with the additional keys that are available on business phones.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>R</th>
<th>S</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>fk1-fk99</td>
<td>Feature Key</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ks</td>
<td>Key State</td>
<td></td>
<td>OO</td>
<td></td>
</tr>
</tbody>
</table>
Feature Key (fk1-fk99)
These events map to all the keys on the phone that are not DTMF keys or locally implemented features (such as volume). The mapping of fk number to key is expected to vary between phones.

Note: Some have suggested parameterizing an fk event, i.e. sending an RQNT with "R: KY/fk" and notifying with "O: KY/fk(1)", but this is problematic. It is desirable to request only the keys that can be pressed in a given state, to eliminate the chance that a mis-pressed button will cancel a timeout signal, and to eliminate message traffic. This is not possible within the confines of MGCP, as requested events cannot be parameterized.

Key State (ks)
This signal is used to indicate the state of a feature key. This signal has 2 parameters: key number and state. The key number maps directly to the feature key number.

The state is a high level description of the state of the key. This allows different phones to implement different indications of state. For example, Phone A may have a multi-color LED associated with feature keys that can blink at different cadences. Phone B might have an LCD beside the keys that can display text or icons. It is up to each phone vendor to determine how to present the state indication.

The following states are used:

<table>
<thead>
<tr>
<th>State</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>enabled</td>
</tr>
<tr>
<td>db</td>
<td>disabled</td>
</tr>
<tr>
<td>id</td>
<td>idle</td>
</tr>
<tr>
<td>dt</td>
<td>dial tone</td>
</tr>
<tr>
<td>cn</td>
<td>connected</td>
</tr>
<tr>
<td>rg</td>
<td>ringing</td>
</tr>
<tr>
<td>rb</td>
<td>Ringback</td>
</tr>
<tr>
<td>ho</td>
<td>holding</td>
</tr>
<tr>
<td>he</td>
<td>held</td>
</tr>
</tbody>
</table>

For example: an RQNT with "S: KY/ks(5,en)" will cause an indicator corresponding to fk5 to indicate that it is enabled. An RQNT with "S: KY/ks(2,rg)" will cause an indicator corresponding to fk2 to indicate that it is ringing.

"en" state
The associated feature will be enabled. Used for keys that turn a feature on or off, such as "Do Not Disturb."

"db" state
The associated feature will be disabled. Used for keys that turn a feature on or off, such as "Do Not Disturb."

"id" state
The specified line appearance is in the idle state,
available for a call.

"dt" state
The specified line appearance is providing dial-tone.

"cn" state
The specified line appearance is actively in a call in the connected state.

"rg" state
The specified line appearance is terminating an incoming call in the ringing state.

"rb" state
The specified line appearance is originating a call in the ringing-back state.

"ho" state
The specified line appearance is in the holding state, with the far end held.

"he" state
The specified line appearance is in the held state, with the far end holding.

Set Label (sl)
This signal is used to set the label on a key. This is used for phones that have an LCD next to the feature keys. It should be accepted but ignored for phones without this capability.

This signal has 2 parameters: key number and label. The key number maps directly to the feature key number. The label is free form text, restricted to the capabilities of the phone.

For example, an RQNT with "S: KY/sl(1,2200)" sets the label next to the fk1 feature key with the extension 2200.

2.2 Business Phone Package

Package Name: BP

The Business Phone Package groups signals other than those related to feature keys and displays.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>R</th>
<th>S</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>hd</td>
<td>Force Offhook</td>
<td></td>
<td>OO</td>
<td></td>
</tr>
<tr>
<td>hu</td>
<td>Force Onhook</td>
<td></td>
<td>OO</td>
<td></td>
</tr>
<tr>
<td>beep</td>
<td>Beep</td>
<td></td>
<td>BR</td>
<td></td>
</tr>
</tbody>
</table>

Force Offhook (hd)
This signal is used to force the phone offhook. If the phone has a speakerphone, it should be activated. This signal can be negated by the user by hanging up.

This can be used if a feature key causes a call to be initiated.
This can also be used for application integration. For example, a user could select a number in an application on their PC, and the phone would be forced offhook and a call initiated.

Force Onhook (hu)
This signal forces the phone onhook. This can be used when the far-end disconnects.

Beep (beep)
Play a beep on the phone.

2.3 Display XML Package

Package Name: XML

The XML Package contains one event/signal that is used to convey XML data to and from the phone.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>R</th>
<th>S</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml</td>
<td>XML Data</td>
<td>x</td>
<td></td>
<td>OO</td>
</tr>
</tbody>
</table>

XML Data (xml)
As an event, if this event is requested in an RQNT with "R: XML/xml", any posts of data from an XML script are returned in an NTFY with "O: XML/xml(post data here)".

As a signal, the parameterized data indicates an URL to an XML script (possibly local), as well as substitution values that depend on the XML script selected. See Section 5 for more information.

3. Endpoint Naming

Because the display state can be somewhat asynchronous from the signaling state of the phone, it is desirable to address the display as a separate MGCP endpoint in order to simplify the call-agent state machine.

For example, suppose a call is presented to the phone, and a display is presented that gives the user the option of redirecting the caller immediately to voice-mail. Selecting the option via the display would cause an XML post to occur, cancelling any timeout signals (the ringing).

In order to simplify the handling of such scenarios, it is recommended that the related display have a different MGCP endpoint name created by inserting a prefix before the endpoint name. The prefix used shall be "disp/".

For example, if the endpoint has the name "ep1@foo.whatever.net", the display would be "disp/ep1@foo.whatever.net".

4. Functions that should be Locally Implemented

There are some functions that should be implemented locally on the endpoint. They are listed in the following sections.
4.1 Volume Control

Volume for ringing, handset, and speakerphone should be implemented locally on the endpoint.

4.2 Audio Path

If the phone includes a speakerphone, activating the speakerphone from the idle state should generate an L/hd event. The user should then be able to switch to handset by lifting the handset, and be able to switch back to speakerphone without any interaction with the call-agent. De-activating the speakerphone with the handset on-hook should generate an L/hu event.

4.3 Microphone mute button and light

If the phone includes a microphone mute button and (optionally) an associated indicator (e.g. light) the functionality of these items should be implemented locally on the endpoint.

5. XML Package Support

Not all business phones have the same display and keypad capabilities. To support these varying devices in a consistent manner, this section outlines an XML framework that is used to drive the phone. In this framework, the call agent pushes XML requests to the endpoints using MGCP signals and events. These XML requests indicate the XML document that is to be rendered on the phone.

5.1 XML Documents

When an XML request is sent to an endpoint, it indicates the XML documents that the endpoint must process. These documents contain tags that are a subset of the Wireless Markup Language (WML) [3] plus some non-WML additions. The tags specify items to be displayed as well as XML events that may be generated as the result of keypad input.

Each XML document, known as a card, defines a user interaction. A group of cards is called a deck. One or more decks define an application. The cards define soft key behavior as well as display behavior, and are mapped to components that implement the behavior of a basic graphical user interface on the display phone. Based on the available requirements, the components needed are:

- **Input box:** allows user input, including editing capabilities, via the keypad.
- **Enumerated list box:** allows the user to select one of a list of items.
- **Itemized list box:** allows the user to select an item using a soft key.
- **Text box:** displays read-only text to the user.
- **Echo box:** displays but does not process user input.

A card may have the following properties.

1. Timed content (e.g. card expiration)
2. Static content (e.g. text)
3. Dynamic content (e.g. call timers/time)
Additionally, cards may also contain variables that may be substituted for values that are specified in an XML request. See the following section for details on variable substitution.

There are cases where the XML scripts handling the display need to use keys that are also used in MGCP. For example, the display could present an enumerated list, where a particular item is selected by pressing the associated number on the dial pad. All user key presses must be routed through the XML component layer.

The XML component layer consumes the key presses or passes them on to the MGCP layer for consumption. The code handling keypresses should present a keypress to the XML code first. If the XML code does not "use" the key, then the key should be presented to the MGCP code. This gives precedence to the XML scripts for keypresses.

5.2 XML Requests

The XML framework uses MGCP as its transport for making requests to the display phone. MGCP is also used to receive asynchronous events from the display phone (e.g. an item has been selected, the user has entered text, etc).

An XML request is made to an endpoint using the XML/xml signal. The signal has the following format:

S: XML/xml(?<url>?</?<card>?<$<variable1>=<value1>?$<variable2>=<value2>?)

The first component of the signal parameter is a URL to the deck. If no scheme is indicated, the file is assumed to be local to the phone. Here are some examples:

ftp://server.company.com/deck1?card1?$var1=val1
http://www.company.com/deck1?card1?$var1=val1
file:///deck1?card1?$var1=val1
dek1?card1?$var1=val1

A card identifier and a list of variable/value pairs follow the URL. The card identifier indicates the card within the deck to display.

The variable/value pairs are substituted into the deck before it is rendered to the display. This means that 1) the variables are deck-scoped and 2) variables not defined in the requested card may be specified in the request as long as they exist in the deck.

For example, a deck may contain the following cards:

```xml
<card id=one>
  <p>$line1</p>
  <timer value="2"/>
  <do type="ontimer">
    <go href="#two/>
  </do>
</card>

<card id=two>
  <p>$line2</p>
</card>
```
And an XML request may look like:

S: XML/xml(deck?one?$line1=abc$line2=xyz)

After variable substitution, the deck will look like:

```xml
<card id=one>
  <p>abc</p>
</card>

<card id=two>
  <p>$line2</p>
</card>
```

Once variable substitution is complete, the card is rendered. If a parameter variable does not exist in the specified card it should be ignored.

When card two is invoked from card1 in response to the timeout action, card two’s variables are substituted with the variables values passed as a request to card one. Card two will look like:

```xml
<card id="two">
  <p>xyz</p>
</card>
```

### 5.3 XML Request History

In order to support navigation through a request history such as when a user cancels a card, the XML layer must maintain a last-in-first-out history of requests made for the endpoint. (See the `<prev>` tag definition in a following section)

### 5.4 XML Events

Whenever the XML layer determines that an event has occurred, it reports the event using the MGCP observed event field:

```xml
O: XML/xml(post?<deck>?<card>?<variable1>=<value1>?<variable2>=<value2>)
```

Here, the event parameter contains the deck and card that generated the event as well as data that is to be processed by the call agent. The data being posted is in the form of a list of variable/value pairs.

In order for the endpoint to properly generate the XML event, it is necessary for the call agent to request the event using the requested events field:

```xml
R: XML/xml
```

This requested event should be combined with the signal request in an RQNT.

### 5.5 XML Tags

Any XML implementation must at a minimum support the XML tags listed in
the table that follows. All tags have a terminator tag of the form 
</tag> to indicate the end of the tag. See the XML grammar in Appendix A.

<table>
<thead>
<tr>
<th>Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;xml&gt;</td>
<td>Marks the beginning of a deck.</td>
</tr>
<tr>
<td>&lt;card&gt;</td>
<td>Marks the beginning of a card</td>
</tr>
<tr>
<td>&lt;p&gt;</td>
<td>Marks the beginning of a paragraph.</td>
</tr>
<tr>
<td>&lt;select&gt;</td>
<td>Defines a list of items that may be selected (an enumerated or itemized list box).</td>
</tr>
<tr>
<td>&lt;option&gt;</td>
<td>Used in conjunction with the &lt;select&gt; tag to specify an individual item that may be selected.</td>
</tr>
<tr>
<td>&lt;input&gt;</td>
<td>Marks the beginning of user input (an input box).</td>
</tr>
<tr>
<td>&lt;echo&gt;</td>
<td>Marks the beginning of an echo box.</td>
</tr>
<tr>
<td>&lt;calltimer&gt;</td>
<td>Call Timer. An incremental timer usually used to maintain the duration of a call.</td>
</tr>
<tr>
<td>&lt;timer&gt;</td>
<td>Card timer. Allows an event to be generated when the timer expires.</td>
</tr>
<tr>
<td>&lt;time&gt;</td>
<td>A tag indicating the current time.</td>
</tr>
<tr>
<td>&lt;do&gt;</td>
<td>Event consumer.</td>
</tr>
<tr>
<td>&lt;go&gt;</td>
<td>Used in conjunction with the &lt;do&gt; tag to indicate a new page to be displayed.</td>
</tr>
<tr>
<td>&lt;prev&gt;</td>
<td>Used in conjunction with the &lt;do&gt; tag to indicate that the previous card in the history should be displayed.</td>
</tr>
</tbody>
</table>

Most of these tags have attributes. Each attribute has one of the following types: String, Time, Enum, Align, Action or URL:

<table>
<thead>
<tr>
<th>Type</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Any string. May not contain any white space (tab, space or newline).</td>
</tr>
<tr>
<td>Time</td>
<td>A string of the format hh:mm:ss where hh indicates the hour (24-hour format), mm indicates the minutes and ss indicates the seconds.</td>
</tr>
<tr>
<td>Enum</td>
<td>Enumeration. A list of acceptable string values.</td>
</tr>
<tr>
<td>Align</td>
<td>Indicates text alignment (left justified, centered or right justified). Valid values are: left, center, right. The default value is: left.</td>
</tr>
<tr>
<td>Action</td>
<td>Defines a string to be sent to the call agent.</td>
</tr>
</tbody>
</table>
This string has the format:
post?%var1=%val1?%var2=%val2
where variables that should be substituted before sending the string to the call agent begin with a ‘%’.
The tags the make up the card determine what variables are available to this string. See the following sections for variables that are defined for each tag.

### URL
The URL may have take several forms:
1. #<card> to indicate another card within the same deck
2. A string of type Action
3. #<prev> to indicate the previous card in the history

### 5.5.1 XML Tag

The `<xml>` tag must be the first tag specified in the deck. It indicates the beginning of the deck.
This tag has no attributes.

### 5.5.2 Card Tag

The `<card>` tag marks the beginning of a new card.

This tag has the following attributes:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Values</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>String</td>
<td>Defines the card identifier. This identifier is referenced in XML requests.</td>
</tr>
</tbody>
</table>

### 5.5.3 P Tag

The `<p>` tag marks the beginning of a new paragraph.

This tag has the following attributes:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Values (default)</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>Enum: wrap/nowrap (wrap)</td>
<td>Specifies whether the paragraph wraps or is truncated when it extends past the display width.</td>
</tr>
<tr>
<td>Align</td>
<td>Align</td>
<td>Specifies alignment of the paragraph.</td>
</tr>
</tbody>
</table>

### 5.5.4 Select Tag

The `<select>` tag marks the beginning of a list of items that may be
selected. Each item is defined using an <option> tag described in the following section.

The tag has the following attributes:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Enum: item/enum (enum)</td>
<td>Specifies the type of list: itemized or enumerated. An itemized list maps options to soft keys.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Specifies name of the list. This attribute is available to any Action string in the card by using the %name variable.</td>
</tr>
<tr>
<td>iname</td>
<td>String</td>
<td>Defines an index variable with the specified name. This variable is used in the &lt;option&gt; tag to specify the index of an item that is selected. The value of this attribute is available to any Action string in the card by using the %iname variable. The value of the index variable is available by using the %&lt;string value&gt; variable. See examples below.</td>
</tr>
</tbody>
</table>

5.5.5 Option Tag

When used in conjunction with the <select> tag, the <option> tag specifies an individual item that may be selected from a list.

The tag has the following attributes:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>String</td>
<td>Defines the value the item. This is used when reporting an event to the call agent. The value of this attribute is available to any Action string in the card by using the %value variable.</td>
</tr>
<tr>
<td>onpick</td>
<td>Action</td>
<td>Defines the string to be sent to the call agent when the item is selected.</td>
</tr>
</tbody>
</table>

5.5.6 Input Tag

The <input> tag specifies that user input is required.
This tag has the following attributes:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Values</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Specifies the name of the input tag. The value of this attribute is available to any Action string in the card by using the %name variable.</td>
</tr>
<tr>
<td>type</td>
<td>Enum: password/text (text)</td>
<td>Specifies whether the input box is in password mode (password) or normal mode (text). When in password mode, user input should be masked.</td>
</tr>
</tbody>
</table>

### 5.5.7 Echo Tag

The `<echo>` tag indicates that user input is required. Any keypad activity is reported to the XML layer but not consumed when this tag is used.

This tag has the following attributes:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Values (default)</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>Enum: on/off (on)</td>
<td>Specifies whether the echo box is in password mode (off) or normal mode (on). When in password mode, user input should be masked.</td>
</tr>
<tr>
<td>align</td>
<td>Align</td>
<td>Specifies the alignment of the echo tag.</td>
</tr>
</tbody>
</table>

### 5.5.8 Calltimer Tag

The `<calltimer>` tag is used to indicate that an incrementing timer is to be displayed.

This tag has the following attributes:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Values</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Time</td>
<td>Specifies the initial value of the call timer.</td>
</tr>
<tr>
<td>align</td>
<td>Align</td>
<td>Specifies the alignment of the call timer.</td>
</tr>
</tbody>
</table>

### 5.5.9 Time Tag

The `<time>` tag is used to display the current time on the phone.

This tag has the following attributes:
5.5.10 Timer Tag

The `<timer>` tag is used to define a timeout for the card. When the timeout occurs, the XML Layer looks for the appropriate `<do>` tag to take appropriate action.

The tag has the following attributes:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Values</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Time</td>
<td>Specifies the initial value of the timer. The timer will decrement the time until it reaches zero at which point the <code>&lt;do&gt;</code> tag is consulted.</td>
</tr>
</tbody>
</table>

5.5.11 Do Tag

The `<do>` tag indicates an action to be performed when the specified event occurs.

Currently, the `<do>` tag can process three events: prev, ontimer and accept. The prev event indicates that the user has requested to cancel the current card.

The ontimer event indicates that the timer defined using the `<timer>` tag has expired.

The accept event indicates that the user has completed inputting from the keypad.

The tag has the following attributes:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Values (default)</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Enum:</td>
<td>Indicates the event on which the tag operates.</td>
</tr>
<tr>
<td></td>
<td>prev/ontimer/accept</td>
<td></td>
</tr>
</tbody>
</table>

5.5.12 Go Tag

The `<go>` tag is used in conjunction with the `<do>` tag to specify a URL to be loaded when the event occurs.

This tag has the following attributes:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Values (default)</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>href</td>
<td>URL</td>
<td>Defines the URL of the next XML page.</td>
</tr>
</tbody>
</table>
The <prev> tag is used in conjunction with the <do> tag to indicate that the previous page in the display history should be rendered. This tag has no attributes.

6. Acknowledgements

Thanks to the following companies and individuals for contributing their experience and thoughts for inclusion in this document.

Arnie Chencinski, Sylantro Systems
Bill Foster, Cisco Systems
Howard Holgate, Cisco Systems
John Weald, Sylantro Systems
Michael Chack, Sylantro Systems
Naga Surendran, Sylantro Systems
Sunil Veluvali, Sylantro Systems

7. References


8. Authors Addresses

Ashok Srinath
Sylantro Systems
910 E. Hamilton Avenue
Campbell, Ca. 95008
Email: Ashok.Srinath@sylantro.com

Gil Levendel
Sylantro Systems
910 E. Hamilton Avenue
Campbell, Ca. 95008
Email: Gil.Levendel@sylantro.com

Kent Fritz
Sylantro Systems
910 E. Hamilton Avenue
Campbell, Ca. 95008
Email: Kent.Fritz@sylantro.com

Raghuraman Kalyanaram
Wipro Systems
Appendix A: BNF description of XML grammar

The parser is case sensitive. In this section we will use the following conventions:

1. Small letters means terminals.
2. Capital strings are non-terminals.
3. \( [A \mid B] \) means either A or B must appear in this place.
4. \( \backslash t, \backslash n, \backslash r, \) blank space are separators.

<table>
<thead>
<tr>
<th>ACTION</th>
<th>:</th>
<th>&lt;go href=&quot;HREFSTRING&quot;/&gt;</th>
<th>&lt;prev/&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIGN</td>
<td>:</td>
<td>Align=[&quot;left&quot; \mid &quot;right&quot;]</td>
<td></td>
</tr>
<tr>
<td>CALLTIMER</td>
<td>:</td>
<td>&lt;calltimer CALLTIMERATTRS/&gt;</td>
<td></td>
</tr>
<tr>
<td>CALLTIMERATTR</td>
<td>:</td>
<td>CALLTIMERATTR CALLTIMERATTRS</td>
<td></td>
</tr>
<tr>
<td>CARDS</td>
<td>:</td>
<td>CARD \mid CARD CARDS</td>
<td></td>
</tr>
<tr>
<td>CARD</td>
<td>:</td>
<td>&lt;card id=STRING&gt; CLUSTERS &lt;/card&gt;</td>
<td></td>
</tr>
<tr>
<td>CARDREFERENCE</td>
<td>:</td>
<td>#STRING</td>
<td></td>
</tr>
<tr>
<td>CLUSTERS</td>
<td>:</td>
<td>CLUSTER \mid CLUSTER CLUSTERS</td>
<td></td>
</tr>
<tr>
<td>CLUSTER</td>
<td>:</td>
<td>CONTROL \mid TIMER \mid ECHO \mid PARAGRAPH COMPONENTS &lt;/p&gt;</td>
<td></td>
</tr>
<tr>
<td>COMPONENTS</td>
<td>:</td>
<td>COMPONENT \mid COMPONENT COMPONENTS</td>
<td></td>
</tr>
<tr>
<td>COMPONENT</td>
<td>:</td>
<td>TEXT \mid INPUTBOX \mid SELECTBOX \mid STIME \mid CALLTIMER</td>
<td></td>
</tr>
<tr>
<td>CONTROL</td>
<td>:</td>
<td>&lt;do CONDITION&gt; ACTION &lt;/do&gt;</td>
<td></td>
</tr>
<tr>
<td>CONDITION</td>
<td>:</td>
<td>type=[&quot;accept&quot; \mid &quot;prev&quot; \mid &quot;ontimer&quot;] label=STRING</td>
<td></td>
</tr>
<tr>
<td>DML</td>
<td>:</td>
<td>&lt;dml id=STRING&gt; CARDS &lt;/dml&gt;</td>
<td></td>
</tr>
<tr>
<td>DIGITS</td>
<td>:</td>
<td>DIGIT \mid DIGIT DIGITS</td>
<td></td>
</tr>
<tr>
<td>DIGIT</td>
<td>:</td>
<td>0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9</td>
<td></td>
</tr>
<tr>
<td>ECHO</td>
<td>:</td>
<td>&lt;echo/&gt; \mid &lt;echo ECHOMODE/&gt;</td>
<td></td>
</tr>
<tr>
<td>ECHOMODE</td>
<td>:</td>
<td>mode=[&quot;on&quot; \mid &quot;off&quot;]</td>
<td></td>
</tr>
<tr>
<td>HREFSTRING</td>
<td>:</td>
<td>CARDREFERENCE \mid POSTSTRING</td>
<td></td>
</tr>
<tr>
<td>INPUTBOX</td>
<td>:</td>
<td>&lt;input INPUTATTRS/&gt;</td>
<td></td>
</tr>
</tbody>
</table>
Appendix B: Sample XML Documents, Renderings and Events

This section presents some sample XML documents and details how they are translated to a business phone with a simple LCD display.
B.1 Sample Deck 1 (Itemized List Box)

Below is a simple deck containing one card that defines a simple main menu interface using an itemized list box:

<xml>
<card id="home">
<p mode="nowrap">$dn <time align="right"></time>
<select type="item" name="Menu" iname="StrMenu">
<option value="1" onpick="post?%deck?%id?%name=%value">MENU</option>
</select>
</p>
</card>
</xml>

The card (home) contains three components:

1. A paragraph (<p>). The paragraph contains a variable ($dn) that determines the phone’s extension.
2. A clock (<time>). The clock is aligned to the right.
3. An itemized list (<select>) containing one item (MENU).

An XML request for this deck and card might look like:

S: XML/xml(deck?home?$dn=2344)

After variable substitution, the phone may render the XML to the display as follows:

-------------
| 2344 11:59 |
| MENU |
-------------
| [XX] [XX] [XX] |

Where MENU maps to the first soft key below the phone’s display. If the user presses the first soft key, the following event will be generated:


B.2 Sample Deck 2 (Enumerated List Box)

The next sample deck defines a simple enumerated list box card:

<xml>
<card id="gelist">
<p>$title
<select name="x-name" iname="x-iname">
<option value="$value1" onpick="post?%deck?%id?%name=%value?%iname=%x-iname">$opt1
</option>
<option value="$value2" onpick="post?%deck?%id?%name=%value?%iname=%x-iname">$opt2
</option>
<option value="$value3" onpick="post?%deck?%id?%name=%value?%iname=%x-iname">$opt3
</option>
</select>
</p>
</card>
</xml>
The card (gelist) contains four components:

1. A paragraph (<p>). The paragraph contains a title variable describing the list contents.
2. An enumerated list (<select>) containing nine items. When an item is selected, the XML layer sends the XML/xml event to the call agent.
3. A do tag (<do>) indicating that when a "previous" event has occurred, to go to the previous page (<prev>).

An XML request for this deck and card might look like:


After variable substitution, the phone may render the XML to the display as follows:

-------------------
<p>| SELECT A CAR      |
| 1. Porsche v      |
| 2. Chevrolet      |</p>
<table>
<thead>
<tr>
<th>3. Toyota</th>
</tr>
</thead>
</table>

Here, the display may be scrolled to reveal the additional items that may be selected and the keypad ‘1’, ‘2’, etc may be used to select the item. These details are phone-specific. For instance, on a larger 4-line display containing navigation keys, the XML may be rendered as follows:

-------------------
<p>| SELECT A CAR      |
| =&gt;Porsche&lt;=       |
| 2. Chevrolet      |</p>
<table>
<thead>
<tr>
<th>3. Toyota v</th>
</tr>
</thead>
</table>

When the user selects item 1, the following message will be sent to the call agent:

O: XML/xml(post?list?gelist?x-name=Item1?x-iname=1)
B.3 Sample Deck 3 (Text Box)

This sample shows how to implement a simple text box:

<xml>
<card id="generic">
<p>$clidpty</p>
<p>CALL FAILED</p>
</card>
</xml>

The card (generic) contains two paragraphs. The absence of a selectable list, input box or echo box indicates that this is a text box.

An XML request for this deck and card might look like:

S: XML/xml(deck?generic?$clidpty=John Doe)

After variable substitution, the phone may render the XML to the display as follows:

-------------------
| JOHN DOE          |
| CALL FAILED       |
-------------------
[XX]  [XX]  [XX]

B.4 Sample Deck 4 (Echo Box)

This sample show how to implement a simple echo box. The XML layer does not consume any keystrokes.

<xml>
<card id="getdigits">
<p>Dial Number:</p>
<echo mode="$mode" align="left"/>
</card>
</xml>

The card (getdigits) contains a paragraph of text and an echo box.

An XML request for this deck and card might look like:

S: XML/xml(deck?getdigits?$mode=on)

After variable substitution, the phone may render the XML to the display as follows:

-------------------
| DIAL NUMBER:      |
|                  |
-------------------
[XX]  [XX]  [XX]

All user input is displayed but not consumed by the XML layer.
### B.5 Sample Deck 5 (Input Box)

This sample implements a basic input box:

```xml
<xml>
<card id="ginput">
<p>$title
<input name="x-name"/>
</p>
<do type="accept">
<go href="post?%deck?%id?%name=%value"/>
</do>
<do type="prev">
<prev></prev>
</do>
</card>
</xml>
```

The card (ginput) contains:
1. A paragraph `<p>`. The paragraph contains a title.
2. An input box `<input>`. The input box consumes keypad events and
   reports them when input is complete.
3. Two event handlers `<do>`. The first handles the accept event. This
   event indicates that the user has completed keypad input and
   posts an observed event to the call agent. The second handles the
   prev event. This event indicates that the user has requested to
   cancel the current card.

An XML request for this deck and card might look like:

```
S: XML/xml(deck?ginput?$title=Enter Digits:)
```

After variable substitution, the phone may render the XML to the
display as follows:

```
---------------------
| ENTER DIGITS:     |
| _                 |
---------------------
 [XX]  [XX]  [XX]
```

It is up to the individual business phone implementation to determine
which soft keys or keypad keys map to such things as "backspace" and
"reset line", etc.

### B.6 Sample Deck 6 (Timers)

To illustrate timers and deck-scoped variable substitution, a two-card
deck is provided:

```xml
<xml>
<card id="connected1">
<timer value="$tvalue"/>
<p mode="nowrap">$cldpty
<select type="item" name="x-name" iname="x-iname">
<option value="TRNSINIT" onpick="post?%deck?%card?%name=%value">TRNS
</option>
<option value="CONFINIT" TRNS>
```

In this example, when the timer expires in card connected1, it generates an ontimer event. This event is consumed by the <do> tag and causes the XML layer to load card with the identifier connected2.

An XML request for these cards might look like:

S: XML/xml(deck?connected1?$tvalue=00:00:05?$cldpty=John Doe?$calltimer=00:00:00)

And might be rendered as:

-------------------
| JOHN DOE          |
| TRNS  CONF  MENU  |
-------------------
[XX]  [XX]  [XX]

Once the timer expires, the XML layer loads the referenced page:

-------------------
| 00:00:05          |
| TRNS  CONF  MENU  |
-------------------
[XX]  [XX]  [XX]
Appendix C: Example usage of MGCP extension packages

C.1 Setting Labels on Phone

Step 1. Call-agent sets labels on several used keys. Should be done at startup. The first 2 keys are line appearance keys. fk8 is a Do Not Disturb function.

```
RQNT 1876 d003@da-003.syltrx.com MGCP 1.0
N: cs@sage.syltrx.com:2427
X: 45
S: KY/sl(1,2315), KY/sl(2,2315), KY/sl(8,DND)
R: KY/fk1, KY/fk2, KY/fk8, KY/fk22, KY/fk23, L/hd
T: L/hu
K: 1873
```

Step 2. Endpoint responds.

```
200 1876 OK
```

C.2 Activating a Feature on a Feature Key

This example shows a feature key that is assigned to "Do Not Disturb" being activated and deactivated.

Step 1. User presses DND key, which is assigned to fk8. Endpoint sends NTFY to call-agent.

```
NTFY 957 d003@da-003.syltrx.com MGCP 1.0
K: 956
N: cs@sage.syltrx.com:2427
X: 45
O: KY/fk8
```

Step 2. Call-agent responds.

```
200 957 OK
```

Step 3. Call-agent sends new RQNT, indicating that DND indicator be activated. Note that the Call-agent also re-sends the state of fk1, which is not actually necessary. The call-agent requests notification of several of the feature keys: fk1 and fk2 are line keys, fk8 is DND, fk22 is redial, and fk23 is Msg.

```
RQNT 2822 d003@da-003.syltrx.com MGCP 1.0
N: cs@sage.syltrx.com:2427
X: 45
S: KY/ks(1,id), KY/ks(8,en)
R: KY/fk1, KY/fk2, KY/fk8, KY/fk22, KY/fk23, L/hd
T: L/hu
K: 2743-2744
```

Step 4. Endpoint responds.

```
200 2822 OK
```

Step 5. User presses DND key again to de-activate DND. Endpoint sends NTFY to call-agent.

    200 958 OK

Step 7. Call-agent sends new RQNT, DND indicator is de-activated.

    RQNT 2823 d003@da-003.syltrx.com MGCP 1.0
    N: cs@sage.syltrx.com:2427
    X: 45
    S: KY/ks(1, id), KY/ks(8, db)
    R: KY/fk1, KY/fk2, KY/fk8, KY/fk22, KY/fk23, L/hd
    T: L/hu
    K: 2822

Step 8. Endpoint responds.

    200 2823 OK

C.3 Generating a Call using Feature Key as a Line Key

This example shows the MGCP messages for dialing an extension after pressing a feature key that is configured as a line appearance key.

Step 1. User presses fk1, which is configured as a line key.

    NTFY 959 d003@da-003.syltrx.com MGCP 1.0
    K: 958
    N: cs@sage.syltrx.com:2427
    X: 45
    O: KY/fk1

Step 2. Call-agent responds.

    200 959 OK

Step 3. Call-agent puts the line key in the "dial tone" state and forces the phone offhook.

    RQNT 2833 d003@da-003.syltrx.com MGCP 1.0
    N: cs@sage.syltrx.com:2427
    X: 45
    S: KY/ks(1, dt), BP/hd
    R: KY/fk1, KY/fk2, KY/fk8, KY/fk22, KY/fk23, L/hd
    T: L/hd
    K: 2823

Step 4. Endpoint responds.

    200 2833 OK


    RQNT 2834 d003@da-003.syltrx.com MGCP 1.0
N: cs@sage.syltrx.com:2427
X: 45
S: L/dl, KY/ks(1,dt)
R: D/[0-9*#T](D), KY/fk1, KY/fk2, KY/fk8, KY/fk22, KY/fk23, L/hu
T: L/hd
D: (*xx|[1-7]xxx|9)

200 2834 OK

Step 7. User dials 2362. Endpoint sends NTFY.

NTFY 960 d003@da-003.syltrx.com MGCP 1.0
K: 959
N: cs@sage.syltrx.com:2427
X: 45
O: D/2,D/3,D/6,D/2

200 960 OK

Step 9. Call-agent puts line in the ringback state. Ring not applied yet.

RQNT 2836 d003@da-003.syltrx.com MGCP 1.0
N: cs@sage.syltrx.com:2427
X: 45
S: KY/ks(1,rb)
R: KY/fk1, KY/fk2, KY/fk8, KY/fk22, KY/fk23, L/hu
T: L/hd
K: 2833, 2834

Step 10. Endpoint responds.
200 2836 OK

Step 11. Call-agent creates connection.

CRCX 2838 d003@da-003.syltrx.com MGCP 1.0
C: 10B
M: RECVONLY

Step 12. Endpoint responds.
200 2838 OK
I: 101
v=0
c= IN IP4 172.16.130.32
m=audio 1108 RTP/AVP 0

Step 13. Call-agent applies ringback.

RQNT 2841 d003@da-003.syltrx.com MGCP 1.0
N: cs@sage.syltrx.com:2427
X: 45
S: KY/ks(1,rb), G/rt
R: KY/fk1, KY/fk2, KY/fk8, KY/fk22, KY/fk23, L/hu

200 2841 OK

Step 15. Call-agent modifies connection.

MDCX 2848 d003@da-003.syltrx.com MGCP 1.0
C: 10B
I: 101
M: SENDRECV
K: 2841-2842

v=0
c=IN IP4 172.16.130.31
m=audio 1124 RTP/AVP 0


200 2848 OK

Step 17. Call-agent puts line in connected state. Added requested events looking for hold (fk21) and conference/transfer (fk24).

RQNT 2849 d003@da-003.syltrx.com MGCP 1.0
N: cs@sage.syltrx.com:2427
X: 45
S: KY/ks(1,cn)
R: KY/fk1, KY/fk2, KY/fk8, KY/fk21, KY/fk24, L/hu
T: L/hd
K: 2842

Step 18. Endpoint responds.

200 2849 OK


DLCX 2873 d003@da-003.syltrx.com MGCP 1.0
C: 10B
I: 101
K: 2848, 2849

Step 20. Endpoint responds.

250 2873 Connection Deleted


RQNT 2876 d003@da-003.syltrx.com MGCP 1.0
N: cs@sage.syltrx.com:2427
X: 45
S: KY/ks(1,id), BP/hu
R: KY/fk1, KY/fk2, KY/fk8, KY/fk21, KY/fk24, L/hu
T: L/hu
K: 2873

Step 22. Endpoint responds.