Network Working Group
Request for Comments: 3149
Category: Informational

A. Srinath
G. Levendel
K. Fritz
Sylantro Systems
R. Kalyanaram
Wipro Systems
September 2001

MGCP Business Phone Packages

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document describes a collection of MGCP (Media Gateway Control Protocol) packages that can be used to take advantage of the feature keys and displays on digital business phones and IP-Phones.

IESG Note

This document is being published for the information of the community. It describes a non-IETF protocol that is currently being deployed in a number of products. Implementers should be aware that the IETF Megaco working group and the ITU-T Study Group 16 have produced a standards track RFC "Megaco Protocol Version 1.0" (RFC 3015, also published as ITU recommendation H.248) which addresses the same problem space and are developing extensions to that protocol for functions of this type.

Table of Contents

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 for only 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, speaker phones and displays, it is necessary to define additional packages as extensions to the MGCP protocol.

These packages, when used in conjunction with the packages currently defined in RFC 2705 (Media Gateway Control Protocol Version 1.0) [1], allow an MGCP Call Agent to control business phone endpoints.

The MGCP extension packages defined here are as follows:

- Feature Key Package

  - Groups 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 to allow mapping of key numbers to features.
    - Key State signal to indicate the state of feature keys.
    - Set Label signal to display a label on the LCD next to a feature key.

- Business Phone Package

  - Groups signals that are not related to feature keys, including:

    - Force Off-hook and Force On-hook signals to allow application integration with speaker phone capabilities.
    - Beep signal to play a beep on the phone.

- Display XML Package

  - Used to convey XML [2] script data to and from the phone to control the display and assign functions to the display soft-keys for event reporting. These include:

- XML event to report user input or selection.
- XML signal to render text to the LCD display.

An MGCP experimental parameter is also defined here:

- User Agent Parameter

  - Used to determine the make and model of a phone

1.1 General Information

A generic business phone typically includes a number of features that provide access to additional functionality useful in a 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:

- Speaker phone microphone and speaker
- Speaker phone button and light
- Messages 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

Examples of fixed buttons functionality are 'hold', 'transfer', 'redial', 'conference', 'call-logs', 'directories', and 'messages'. Fixed buttons may vary from phone to phone. While the packages described here would allow these to be reported to a Call Agent, the Call Agent would also need to determine which feature key number corresponds to a particular pre-assigned function.

Since MGCP assumes a call control architecture where the call control "intelligence" is outside the Gateways and handled by external call control elements, the programming of the feature keys would be resident in the Call Agent. If the user were to press the 'hold' button, the phone would simply report the key number, and the burden

of recognizing that this feature key is assigned to the 'hold' function, and providing such functionality, is left to the Call Agent.

1.2 Objectives

The high level objectives that were considered in generating the packages 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 the flexibility to accommodate a variety of application needs and the 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.

Package	Name	Defined
Generic Media Package	G	in RFC 2705
DTMF package	D	in RFC 2705
Line Package	L	in RFC 2705
Handset Package	H	in RFC 2705
Feature Key Package	KY	in this spec
Business Phone Package	BP	in this spec
Display XML Package	XML	in this spec

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 if the event can be requested by the Call Agent.

S: if nothing appears in this column for an event, then the event cannot be signalled on command by the Call Agent. Otherwise, the following symbols identify the type of signal:

OO On/Off signal. The signal is turned on until requested 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 events and signals that are associated with the additional keys that are available on business phones.

Symbol	Definition	R	S	Duration
fk1-fk99	Feature Key	x		
ks	Key State		OO	
ls	Set Label		OO	

Feature Key (fk1-fk99)

These events map to all the keys on the phone that are not DTMF keys or locally implemented functions (such as volume). The mapping of fk number to key is expected to vary between phones.

Note: Some have suggested parameterizing the fk event, i.e., sending an RQNT with "R: KY/fk" and an NTFY 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, as well as to reduce 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. It should be ignored by phones without this capability.

This signal has two 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:

State	Definition
en	enabled
db	disabled
id	idle
dt	dial tone
cn	connected
dc	disconnected
rg	ringing
rb	ringback
ho	holding
he	held

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 is enabled. Used for keys that turn a feature on or off, such as "Do Not Disturb."

"db" state

The associated feature is 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.

"dc" state

The specified line appearance is disconnected, but the corresponding line is still active (the user is still offhook).

"rg" state

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

"rb" state

The specified line appearance is originating an outgoing 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 (ls)

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 ignored by 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/ls(1,2200)" sets the label next to the fk1 feature key to the string "2200" (a phone extension).

2.2 Business Phone Package

Package Name: BP

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

Symbol	Definition	R	S	Duration
hd	Force Offhook		OO	
hu	Force Onhook		OO	
beep	Beep		BR	

Force Offhook (hd)

This signal is used to force the phone offhook. If the phone has a speaker phone, 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. See the sample call flow in Appendix C.

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, or if a feature key causes a call to be terminated.

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.

Symbol	Definition	R	S Duration
xml	XML Data	x	OO

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 a 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 and Phone Type Determination

Because the display state can be asynchronous from the signaling state of the phone, it is desirable to address the display as a separate MGCP endpoint.

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 expected that the related display have a different MGCP endpoint name, created by inserting a prefix before the phone endpoint name. The prefix used shall be "disp/".

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

The Call Agent must be able to determine which feature key number corresponds to a particular pre-assigned function. For example, one phone may have the pre-assigned functions of 'redial' and 'hold' mapped to feature keys numbered fk1 and fk23, respectively. Another phone may not report fk23 at all, and have the pre-assigned function of 'transfer' mapped to fk1. Also, since the programming of feature keys would be resident in the Call Agent, a user-interface that

allows the programming of these keys must know the keys supported on the phone, in order for the Call Agent to request the appropriate feature keys.

Determination of such basic capabilities must occur at the moment when the phone sends its first RSIP message to a Call Agent. While it might be possible to define packages with events and signals that allow for an exhaustive discovery of the layout of a particular phone, a simpler and more reasonable approach would be for the Call Agent to discover the make and model of the phone, and thus determine the capabilities of the phone. To this end, an experimental parameter, "X-UA" has been introduced for use in the Requested-Info field (F:) of the AUEP method. The response to the "X-UA" is expected to be a string that uniquely identifies the make and model of the phone. Note that per RFC 2705, a Gateway must ignore experimental parameters prefixed as "X-" that it cannot support, versus respond with an error code such as 511 (Unrecognized extension). See the sample call flow in Appendix C.

4. Functions that should be Locally Implemented

Some functions should be implemented locally on the Gateway. These are listed in the following sections.

4.1 Volume Control

Volume for ringing, handset, and speaker phone should be implemented locally on the Gateway.

4.2 Audio Path

If the phone includes a speaker phone, activating the speaker phone from the idle state should generate an offhook (L/hd) event. The user should then be able to switch to handset mode by lifting the handset, and be able to switch back to speaker phone mode without any interaction with the Call Agent. De-activating the speaker phone with the handset on-hook should generate an onhook (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 Gateway.

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 Gateway using MGCP signals. These XML requests indicate the XML document that is to be rendered on the phone.

When a user inputs data or makes a selection from a display, the Gateway "posts" an XML request to the Call Agent using MGCP events.

5.1 XML Documents

When an XML signal 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. These tags specify items to be displayed as well as XML events that may be reported as the result of user 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 to be substituted for values that are specified in an XML request. See section 5.2 for details on variable substitution.

There are cases where the XML scripts handling the display need to use keys that are also used by the phone. 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 display layer either consumes the key presses or passes them on to the phone layer for consumption.

The code handling key presses should thus present a key press to the display code first. If the display code does not "use" the key press, then the key press should be presented to the phone code. This gives precedence to the XML scripts for key presses.

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, or the user has entered text).

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 deck 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?$var1=val1
deck1?card1?$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 the variables are deck-scoped, and variables not defined in the requested card must be populated in other cards in the same deck if defined therein.

For example, a deck may contain the following cards:

```
<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:

```
<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 anywhere in the deck 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:

```
<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 section 5.5.13.)

5.4 XML Events

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

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 Gateway to properly generate the XML event, it is necessary for the Call Agent to request the event using the requested events field:

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.

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

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

Type	Format
String	Any string. May not contain any white spaces (tab, space or newline).
Time	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.
Enum	Enumeration. A list of acceptable string values.
Align	Indicates text alignment (left justified, centered or right justified). Valid values are: left, center, right. The default value is: left.
Action	Defines a string to be sent to the Call Agent. 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 that 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:

Attribute Name	Values	Usage
Id	String	Defines the card identifier. This identifier is referenced in XML requests.

5.5.3 P Tag

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

This tag has the following attributes:

Attribute Name	Values (default)	Usage
Mode	Enum: wrap/nowrap (wrap)	Specifies whether the paragraph wraps or is truncated when it extends past the display width.
Align	Align	Specifies alignment of the paragraph.

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 section 5.5.5.

This tag has the following attributes:

Attribute Name	Values (default)	Usage
type	Enum: item/enum (enum)	Specifies the type of list: itemized or enumerated. An itemized list maps options to soft keys.
name	String	Specifies name of the list. This attribute is available to any Action string in the card by using the %name variable.
iname	String	Defines an index variable with the specified name. This variable is used in the <option> 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 %<string value> variable. See examples below.

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.

This tag has the following attributes:

Attribute Name	Values	Usage
value	String	Defines the value of 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.
onpick	Action	Defines the string to be sent to the Call Agent when the item is selected.

5.5.6 Input Tag

The <input> tag specifies that user input is required.

This tag has the following attributes:

Attribute Name	Values	Usage
name	String	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.
type	Enum: password/text (text)	Specifies whether the input box is in password mode (password) or normal mode (text). When in password mode, user input should be masked.

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:

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

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:

Attribute Name	Values	Usage
value	Time	Specifies the initial value of the call timer.
align	Align	Specifies the alignment of the call timer.

5.5.9 Time Tag

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

This tag has the following attributes:

Attribute Name	Values	Usage
align	Align	Specifies the alignment of the time.

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.

This tag has the following attributes:

Attribute Name	Values	Usage
Value	Time	Specifies the initial value of the timer. The timer will decrement the time until it reaches zero at which point the <do> tag is consulted.

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.

This tag has the following attributes:

Attribute Name	Values (default)	Usage
Type	Enum: prev/ontimer/accept	Indicates the event on which the tag operates.

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:

Attribute Name	Values (default)	Usage
href	URL	Defines the URL of the next XML page.

5.5.13 Prev Tag

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. Security Considerations

This extension introduces no new security considerations beyond those discussed in RFC 2705 [1].

7. 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

8. References

[1] Arango, M., Dugan A., Elliot, I., Huitema, C. and S. Pickett, "Media Gateway Control Protocol (MGCP)" RFC 2705, October 1999.

[2] Bray, T., Paoli, J. and C. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0", W3C Proposed Recommendation, February 10, 1998.

[3] "Wireless Application Protocol Wireless Markup Language Specification Version 1.2", WAP Forum, November 1999.

9. 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
Keonics Electronic City
Hosur Road, Bangalore-561 229, India

EMail: Raghuraman.Kal@wipro.com

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 | B] means either A or B must appear in this place.
4. \t, \n, \r, blank space are separators.

ACTION	:	<go href="HREFSTRING"/> | <prev/>
ALIGN	:	Align=["left" | "right"]
CALLTIMER	:	<calltimer CALLTIMERATTRS/>
CALLTIMERATTRS	:	CALLTIMERATTR | CALLTIMERATTR CALLTIMERATTRS
CALLTIMERATTR	:	value=STRING | ALIGN
CARDS	:	CARD | CARD CARDS
CARD	:	<card id=STRING> CLUSTERS </card>
CARDREFERENCE	:	#STRING
CLUSTERS	:	CLUSTER | CLUSTER CLUSTERS
CLUSTER	:	CONTROL | TIMER | ECHO | PARAGRAPH COMPONENTS </p>
COMPONENTS	:	COMPONENT | COMPONENT COMPONENTS
COMPONENT	:	TEXT | INPUTBOX | SELECTBOX | STIME | CALLTIMER
CONTROL	:	<do CONDITION> ACTION </do>
CONDITION		type=["accept" | "prev" | "ontimer"] label=STRING | type=["accept" | "prev" |"ontimer"]
DIGITS	:	DIGIT | DIGIT DIGITS
DIGIT	:	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
DECK	:	<xml id=STRING> CARDS </xml>
ECHO	:	<echo/> | <echo ECHOMODE/>

ECHOMODE	:	mode=["on" | "off"]
HREFSTRING	:	CARDREFERENCE | POSTSTRING
INPUTBOX	:	<input INPUTATTRS/>
INPUTATTRS	:	INPUTATTR | INPUTATTR INPUTATTRS
INPUTATTR	:	name=STRING | type=["text" | "password"] | value=STRING
NAMEVALUES	:	NAMEVALUE | NAMEVALUE?NAMEVALUES
NAMEVALUE	:	NAMEVALUEELEM | NAMEVALUEELEM=NAMEVALUELEM
NAMEVALUELEM	:	%TEXT | TEXT
OPTIONS	:	OPTION | OPTION OPTIONS
OPTION	:	<option value=STRING onpick=HREFSTRING> TEXT </option>
PARAGRAPH	:	<p TXTFORMAT> | <p>
POSTSTRING	:	post?%deck?%id?NAMEVALUES | post?NAMEVALUES
SELECTBOX	:	<select SELECTATTRS> OPTIONS </select>
SELECTATTRS	:	SELECTATTR | SELECTATTR SELECTATTRS
SELECTATTR	:	name=STRING | iname=STRING | type="item"
STIME	:	<time STIMEATTRS/>
STIMEATTRS	:	STIMEATTR | STIMEATTR STIMEATTRS
STIMEATTR	:	value=STRING | format=STRING | ALIGN
STRING	:	Any string enclosed in a pair of quotes ("")
TEXT	:	TEXTELEM | TEXTELEM TEXT
TEXTELEM	:	any string outside of the < .. > and which consists of any symbols except '<' and '\n'
TIMER	:	<timer value="DIGITS"/>

TXTFORMAT	:	ALIGN | TXTMODE | ALIGN TXTMODE | TXTMODE ALIGN
TXTMODE	:	mode=["wrap" | "nowrap"]
		\t, \n, \r, blank space are separators.

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 shows 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]
```

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

O: XML/xml(post?basic?home?Menu=1).

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">$opt3
</option>
<option value="$value4"
onpick="post?%deck?%id?%name=%value?%iname=%x-iname">$opt4
</option>
<option value="$value5"
onpick="post?%deck?%id?%name=%value?%iname=%x-iname">$opt5
</option>
</select>
</p>
<do type="prev">
<prev></prev>
</do>
</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 five 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:

```
S: XML/xml(list?gelist?$title=Select a Car?
$value1=Item1?$opt1=Porsche?
$value2=Item2?$opt2=Chevrolet?
$value3=Item3?$opt3=Toyota?
$value4=Item4?$opt4=Daewoo?
$value5=Item5?$opt5=Yugo)
```

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

```
--------------------
|SELECT A CAR      |
|1. Porsche       v|
--------------------
  [XX]  [XX]  [XX]
```

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:

```
--------------------
|SELECT A CAR      |
|=>Porsche<=       |
|  Chevrolet       |
|  Toyota         v|
--------------------
```

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:

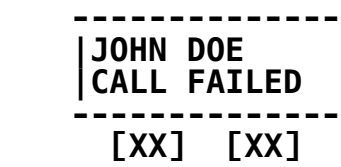
```
<xml>
<card id="generic">
<p>$cldpty</p>
<p>CALL FAILED</p>
</card>
</xml>
```

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?$cldpty=John Doe)

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



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.

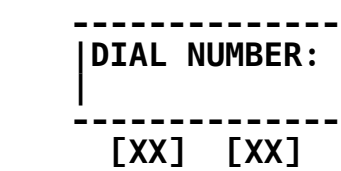
```
<xml>
<card id="getdigits">
<p>Dial Number:</p>
<echo mode="$mode" align="left"/>
</card>
</xml>
```

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:



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>
<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>
```

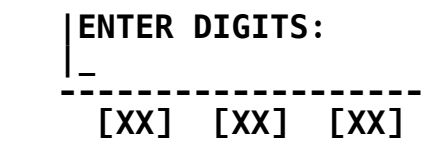
The card (ginput) contains:

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 revert back to the previous 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:



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

B.6 Sample Deck 6 (Timers)

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

```
<xml>
<card id="connected1">
<timer value="$tvalue"/>
<p mode="nowrap">$cldpty
<select type="item" name="x-name" iname="x-iname">
<option value="1"
onpick="post?TRNSINIT">TRNS
</option>
<option value="2"
onpick="post?CONFINIT">CONF
</option>
<option value="3"
onpick="post?%deck?%card?%name=%value">MENU
</option>
</select>
</p>
<do type="ontimer">
<go href="#connected2"/>
</do>
</card>

<card id="connected2">
<p mode="nowrap">
<calltimer value="$calltimer" align="right"/>
<select type="item" name="x-name">
<option value="1"
onpick="post?TRNSINIT">TRNS
</option>
<option value="2"
onpick="post?CONFINIT">CONF
</option>
<option value="3"
onpick="post?%deck?%card?%name=%value" >MENU
</option>
</select>
</p>
</card>
</xml>
```

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 a 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/ls(1,2315), KY/ls(2,2315), KY/ls(8,DND)
R: KY/fk1, KY/fk2, KY/fk8, KY/fk22, KY/fk23, L/hd
T: L/hu
K: 1873
```

Step 2. Gateway responds.

```
200 1876 OK
```

C.2 Activating a Function 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. Gateway 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 messages.

```
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. Gateway responds.

```
200 2822 OK
```

Step 5. User presses DND key again to de-activate DND. Gateway sends NTFY to Call Agent.

```
NTFY 958 d003@da-003.syltrx.com MGCP 1.0
K: 957
N: cs@sage.syltrx.com:2427
X: 45
O: KY/fk8
```

Step 6. Call Agent responds.

```
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. Gateway responds.

```
200 2823 OK
```

C.3 Generating a Call using a 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/hu
T: L/hd
K: 2823
```

Step 4. Gateway responds.

```
200 2833 OK
```

Step 5. Call Agent applies dial-tone.

```
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)
```

Step 6. Gateway responds.

```
200 2834 OK
```

Step 7. User dials 2362. Gateway 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
```

Step 8. Call Agent responds.

```
200 960 OK
```

Step 9. Call Agent puts line in the ringback state. Ringback 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. Gateway 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. Gateway responds.

```
200 2838 OK
I: 101

v=0
o=- 998557784 998557784 IN IP4 38.187.114.41
s=MGCP RTP Session
c=IN IP4 172.16.130.32
t=0 0
m=audio 1108 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

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
T: L/hd
```

Step 14. Gateway responds.

```
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
o=- 7960 7960 IN IP4 38.187.114.215
s=MGCP Call
c=IN IP4 172.16.130.31
t=0 0
m=audio 1124 RTP/AVP 0
```

Step 16. Gateway responds.

```
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. Gateway responds.

```
200 2849 OK
```

Step 19. Far end disconnects. Call Agent deletes connection.

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

Step 20. Gateway responds.

```
250 2873 Connection Deleted
```

Step 21. Call Agent forces endpoint onhook/idle.

```
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/fk22, KY/fk23, L/hd
T: L/hu
K: 2873
```

Step 22. Gateway responds.

```
200 2876 OK
```

C.4 Determining the Make and Model of a Phone

Step 1. Gateway restarts.

```
RSIP 1 *@alpha175.sylantro.com MGCP 1.0
RM: restart
```

Step 2. Call Agent responds.

```
200 1 OK
```

Step 3. Call Agent audits the Gateway to determine list of endpoints

```
AUEP 1000 *@alpha175.sylantro.com MGCP 1.0
```

Step 4. Gateway responds.

```
200 1000 OK
Z: a004@alpha175.sylantro.com
Z: d001@alpha175.sylantro.com
Z: d002@alpha175.sylantro.com
Z: d003@alpha175.sylantro.com
```

Step 5. For each endpoint, Call Agent determines capabilities and user-agent (phone-type)

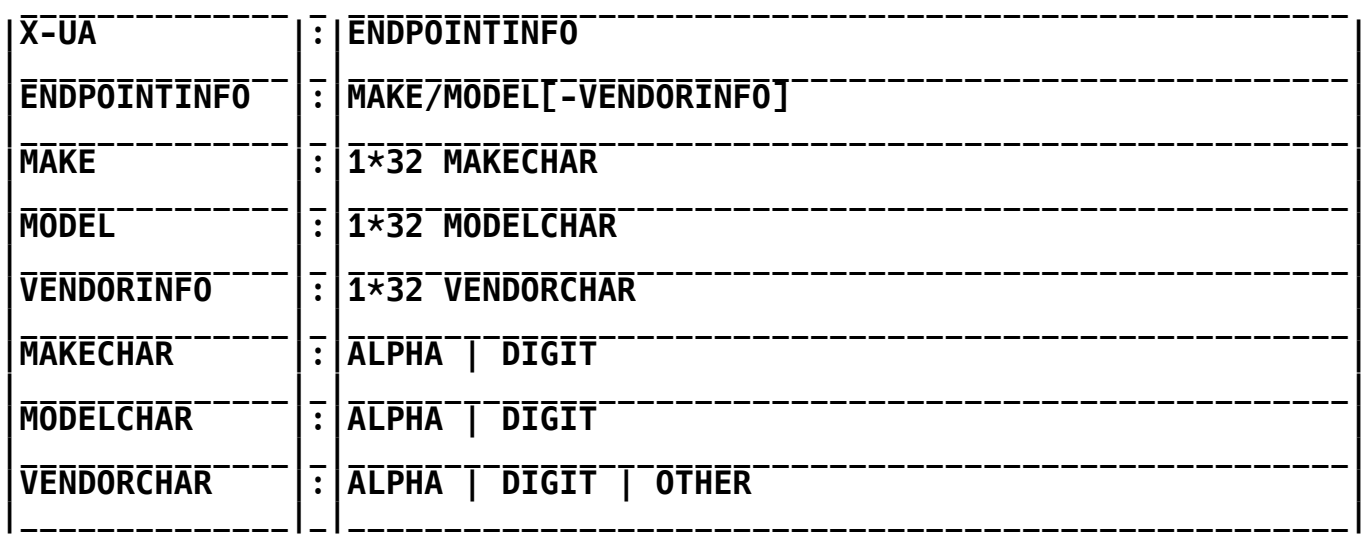
```
AUEP 1040 d003@alpha175.sylantro.com MGCP 1.0
K: 1039
F: A,X-UA
```

Step 6. Gateway responds.

```
200 1040 OK
A: v:D;L;KY;X-BP;G;BP
X-UA: Sylantro/DKT2010-CA204#CA010
```

Appendix D: BNF Description of X-UA Parameter

Since parts of the X-UA parameter must be parseable in order for a Call Agent to treat similar phones in a similar manner, a formal grammar for this parameter is provided.



X-UA	:	ENDPOINTINFO
ENDPOINTINFO	:	MAKE/MODEL[-VENDORINFO]
MAKE	:	1*32 MAKECHAR
MODEL	:	1*32 MODELCHAR
VENDORINFO	:	1*32 VENDORCHAR
MAKECHAR	:	ALPHA | DIGIT
MODELCHAR	:	ALPHA | DIGIT
VENDORCHAR	:	ALPHA | DIGIT | OTHER

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.