Network Working Group
Request for Comments: 2897
Category: Informational

D. Cromwell
Nortel Networks

August 2000

Proposal for an MGCP Advanced Audio Package

Status of this Memo

Copyright Notice

Abstract

This document is a proposal to add a new event/signal package to the MGCP (Media Gateway Control Protocol) protocol to control an ARF (Audio Resource Function) which may reside on a Media Gateway or specialized Audio Server.

This event package provides support for the standard IVR (Interactive Voice Response) operations of PlayAnnouncement, PlayCollect, and PlayRecord. It supports direct references to simple audio as well as indirect references to simple and complex audio. It provides audio variables, control of audio interruptibility, digit buffer control, special key sequences, and support for reprompting during data collection. It also provides an arbitrary number of user defined qualifiers to be used in resolving complex audio structures. For example, the user could define qualifiers for any or all of the following: language, accent, audio file format, gender, speaker, or customer.

Table of Contents

1. Introduction

The following syntax supports both simple and complex audio structures. A simple audio structure might be a single announcement such as "Welcome to Bell South's Automated Directory Assistance Service". A more complex audio structure might consist of an announcement followed by voice variable followed by another announcement, for example "There are thirty seven minutes remaining on your prepaid calling card," where "There are" is a prompt, the number of minutes is a voice variable, and "minutes remaining on your prepaid calling card" is another prompt.

It is also possible to define complex audio structures that are qualified by user defined selectors such as language, audio file format, gender, accent, customer, or voice talent. For instance, if the above example were qualified by language and accent selectors, it would be possible to play "There are thirty seven minutes remaining on your prepaid calling card" in English spoken with a southern accent or in English spoken with a mid-western accent, providing that the audio to support this had been provisioned.

There are two methods of specifying complex audio. The first is to directly reference the individual components. This requires a complete description of each component to be specified via the protocol. The second method is to provision the components on the Audio Server as a single entity and to export a reference to that entity to the call agent. In this case, only the reference (plus any

dynamic data required, such as a variable data) is passed via the protocol, and no specification of individual components is necessary.

The Audio Server Package provides significant functionality most of which is controlled via protocol parameters. Most parameters are optional, and where ever possible default to reasonable values. An audio application that references to provisioned, complex audio structures, and which takes advantage of parameter optionality and defaults, can specify audio events using a minimum of syntax.

1.1. Background

The next two sections contain background information which may be helpful in understanding the syntax.

1.1.1. Sequence And Sets

The syntax supports abstractions of set and sequence for storing and referencing audio data.

A sequence is a provisioned sequence of one or more audio segments. Component segments are not necessarily all of the same type. Every sequence is assigned a unique segment id. On playback, a sequence id reference is deconstructed into its individual parts, each of which is played in order.

A set is a provisioned collection of audio segments with an associated selector. On playback, the selector value is resolved to a particular set element. Selector types are supported by the syntax, but individual selector types are not defined in the syntax except for the pre-defined language selector; they are instead defined by the user (i.e. provisioner). A user could define one or more of the following selector types: language, accent, audio file format, gender, accent, customer, or day of the week. For each selector type, the user must define a range of valid values. The user may also choose to define a default value. At runtime if a selector value is not supplied the default value is used.

For example, to support an application which plays a particular piece of audio in either English, French, or Russian, a provisioner would define a set with the pre-defined selector, "Lang", and would define three possible values for that selector, "eng", "fra", and "rus". The provisioner would then provision three recordings of the prompt, one in each language, and would associate the French recording with the "fra" selector value, etc. The provisioner also could define a default value of the selector when no selector value is supplied, "eng" for instance. The entire set would be assigned a unique segment id.

At runtime a reference to the set with the selector set to "rus" would result in the Russian version of the prompt being played. A reference to the set with no selector would result in the English version of the prompt being played since English has been set as the default selector value.

Nested definition of both sets and sequences is allowed, i.e. it legal to define a set of sets or a sequence of sequences. In addition, audio structures may also be specified by intermixing sets and sequences, and it is possible to specify a set of sequences or a sequence containing one or more set elements. Direct or transitive definition of a set or segment in terms of itself is not allowed.

1.1.2. Segment Types

The syntax supports the following segment types:

RECORDING: A reference by unique id to a single piece of recorded audio.

RECORDINGs may be provisioned or they may be made during the course of a call. A RECORDING made during the course of a call can be temporary or persistent. A temporary RECORDING lasts only for the life of the call during which it was recorded. A persistent RECORDING lasts beyond the live of the call during which it was recorded.

A provisioned RECORDING may be replaced (or overridden) by a persistent RECORDING. A reference to the id of the provisioned RECORDING will then resolve to the persistent RECORDING. The overriding persistent audio can subsequently be deleted and the original provisioned audio can be restored.

A provisioned RECORDING may be overridden more than once. In this case, the id of the provisioned RECORDING refers to the latest overriding RECORDING. When the overriding RECORDING is deleted, the original provisioned RECORDING is restored, even if the segment has been overridden multiple times.

TEXT: A reference to a block of text to be converted to speech or to be displayed on a device. Reference may be by unique id to a block of provisioned text or by direct specification of text in a parameter.

SILENCE: A specification of a length of silence to be played in units of 100 milliseconds.

TONE: The specification of a tone to be played by algorithmic generation. Most tones however will probably be recorded, not generated. Exact specification of this segment type is tbd.

VARIABLE: The specification of a voice variable by the parameters of type, subtype, and value. Specification of variables is considered in more detail in a subsequent section of this document.

SEQUENCE: A reference by unique id to a provisioned sequence of mixed RECORDING, TEXT, SILENCE, TONE, VARIABLE, SET, or SEQUENCE segments. Nested definition of SEQUENCE segments is allowed. Direct or transitive definition of a SEQUENCE segment in terms of itself is not allowed.

SET: A reference by unique id to a provisioned set of segments. The intended and recommended use of the SET type is that all segments in the set should be semantically equivalent, however there is no real way of enforcing this restriction either in the protocol or in provisioning. Every set has an associated selector which is used at runtime to resolve the set reference to a specific element of the set. The elements of a set may one of the following segment types: RECORDING, TEXT, TONE, SILENCE, SEQUENCE, or SET. Specific selector types are not specified by the protocol and must be defined by the user. Nested definition of SET segments is allowed. Direct or transitive definition of a SET segment in terms of itself is not allowed.

2. Advanced Audio Package

Package Name: AU

This package defines events and signals for an ARF package for an Audio Server Media Gateway.

3. Events

Symbol	Definition	R	S	Duration
pa(parms)	PlayAnnouncement		TO	variable
pc(parms)	PlayCollect		TO	variable
pr(parms)	PlayRecord		TO	variable
es(parm)	EndSignal		BR	
oc(parms)	OperationComplete	x		
of(parms)	OperationFailed	x		

The events provided by the AS Package are defined as follows:

PlayAnnouncement:
Plays an announcement in situations where there is no need for interaction with the user. Because there is no need to monitor the incoming media stream this event is an efficient mechanism for treatments, informational announcements, etc.

PlayCollect:
Plays a prompt and collects DTMF digits entered by a user. If no digits are entered or an invalid digit pattern is entered, the user may be reprompted and given another chance to enter a correct pattern of digits. The following digits are supported: 0-9, *, #, A, B, C, D. By default PlayCollect does not play an initial prompt, makes only one attempt to collect digits, and therefore functions as a simple Collect operation. Various special purpose keys, key sequences, and key sets can be defined for use during the PlayCollect operation.

PlayRecord:
Plays a prompt and records user speech. If the user does not speak, the user may be reprompted and given another chance to record. By default PlayRecord does not play an initial prompt, makes only one attempt to record, and therefore functions as a simple Record operation.

OperationComplete:
Detected upon the successful completion of a Play, PlayRecord, or Play Collect signal.

OperationFailed:
Detected upon the failure of a Play, PlayRecord, or PlayCollect signal.

EndSignal:
Gracefully terminates a Play, PlayCollect, or PlayRecord signal. For each of these signals, if the signal is terminated with the EndSignal signal the resulting OperationComplete event or OperationFailed event will contain all the parameters it would normally, including any collected digits or the recording id of the recording that was in progress when the EndSignal signal was received.

4. Signal Interactions

If an Advanced Audio Package signal is active on an endpoint and another signal of the same type is applied, the two signals including parameters and parameter values will compared If the signals are identical, the signal in progress will be allowed to continue and the new signal will be discarded. Because of this behavior the Advanced Audio Package may not interoperate well with some other packages such as the Line and Trunk packages.

5. Parameters

The PlayAnnouncement, PlayRecord, and PlayCollect events may each be qualified by a string of parameters, most of which are optional. Where appropriate, parameters default to reasonable values. The only event with a required parameter is PlayAnnouncement. If a Play-Announcement event is not provided with a parameter specifying some form of playable audio an error is returned to the application.

These parameters are shown in the following table:

Parameters				
Symbol	Definition	pl	pc	pr
an	announcement	x		
ip	initial prompt		x	x
rp	reprompt		x	x
nd	no digits reprompt		x	
ns	no speech reprompt			x
fa	failure announcement		x	x
sa	success announcement		x	x
ni	non-interruptible play		x	x
it	iterations	x		
iv	interval	x		
du	duration	x		
sp	speed	x	x	x
vl	volume	x	x	x
cb	clear digit buffer		x	x
mx	maximum # of digits		x	
mn	minimum # of digits		x	
dp	digit pattern		x	
fdt	first digit timer		x	
idt	inter digit timer		x	
edt	extra digit timer		x	
prt	pre-speech timer			x
pst	post-speech timer			x
rlt	total recording length timer			x
rsk	restart key		x	x
rik	reinput key		x	x
rtk	return key		x	x
psk	position key		x	x
stk	stop key		x	x
sik	start input key		x	
eik	end input key		x	x
iek	include end input key		x	
na	number of attempts		x	x

Parameters to the Advanced Audio Package events are defined as follows:

Announcement:

An announcement to be played. Consists of one or more audio segments.

Initial Prompt:

The initial announcement prompting the user to either enter DTMF digits or to speak. Consists of one or more audio segments. If not specified (the default), the event immediately begins digit collection or recording.

Reprompt:

Played after the user has made an error such as entering an invalid digit pattern or not speaking. Consists of one or more audio segments. Defaults to the Initial Prompt.

No Digits Reprompt:

Played after the user has failed to enter a valid digit pattern during a PlayCollect event. Consists of one or more audio segments. Defaults to the Reprompt.

No Speech Reprompt:

Played after the user has failed to speak during a PlayRecord event. Consists of one or more audio segments. Defaults to the Reprompt.

Failure Announcement:

Played when all data entry attempts have failed. Consists of one or more audio segments. No default.

Success Announcement:

Played when data collection has succeeded. Consists of one or more audio segments. No default.

Non-Interruptible Play:

If set to true, initial prompt is not interruptible by either voice or digits. Defaults to false. Valid values are the text strings "true" and "false".

Iterations:

The maximum number of times an announcement is to be played. A value of minus one (-1) indicates the announcement is to be repeated forever. Defaults to one (1).

Interval:

The interval of silence to be inserted between iterative plays. Specified in units of 100 milliseconds. Defaults to 10 (1 second).

Duration:

The maximum amount of time to play and possibly replay an announcement. Takes precedence over iteration and interval. Specified in units of 100 milliseconds. No default.

Speed:

The relative playback speed of announcement specifiable as a positive or negative percentage of the original playback speed.

Volume:

The relative playback volume of announcement specifiable as a positive or negative decibel variation from the original play-back volume.

Clear Digit Buffer:

If set to true, clears the digit buffer before playing the initial prompt. Defaults to false. Valid values are the text strings "true" and "false".

Maximum # Of Digits:

The maximum number of digits to collect. Defaults to one. This parameter should not be specified if the Digit Pattern parameter is present.

Minimum # Of Digits:

The minimum number of digits to collect. Defaults to one. This parameter should not be specified if the Digit Pattern parameter is present.

Digit Pattern:

A legal digit map as described in section 7.1.14 of the Megaco protocol [6] using the DTMF mappings associated with the Megaco DTMF Detection Package described in the Megaco protocol document

[6]. This parameter should not be specified if one or both of the Minimum # Of Digits parameter and the Maximum Number Of Digits parameter is present.

First Digit Timer:

The amount of time allowed for the user to enter the first digit. Specified in units of 100 milliseconds. 50 (5 seconds).

Inter Digit Timer:

The amount of time allowed for the user to enter each subsequent digit. Specified units of 100 milliseconds seconds. Defaults to 30 (3 seconds).

Extra Digit Timer:

The amount of time to wait for a user to enter a final digit once the maximum expected amount of digits have been entered. Typically this timer is used to wait for a terminating key in applications where a specific key has been defined to terminate input. Specified in units of 100 milliseconds. If not specified, this timer is not activated.

Pre-speech Timer:

The amount of time to wait for the user to initially speak. Specified in units of 100 milliseconds. Defaults to 30 (3 seconds).

Post-speech Timer:

The amount of silence necessary after the end of the last speech segment for the recording to be considered complete. Specified in units of 100 milliseconds. Defaults to 20 (2 seconds).

Recording Length Timer:

The maximum allowable length of the recording, not including pre or post speech silence. Specified in units of 100 milliseconds. This parameter is mandatory.

Restart Key:

Defines a key sequence consisting of a command key optionally followed by zero or more keys. This key sequence has the following action: discard any digits collected or recording in progress, replay the prompt, and resume digit collection or

recording. No default. An application that defines more than one command key sequence, will typically use the same command key for all command key sequences. If more than one command key sequence is defined, then all key sequences must consist of a command key plus at least one other key.

Reinput Key:

Defines a key sequence consisting of a command key optionally followed by zero or more keys. This key sequence has the following action: discard any digits collected or recordings in progress and resume digit collection or recording. No default. An application that defines more than one command key sequence, will typically use the same command key for all command key sequences. If more than one command key sequence is defined, then all key sequences must consist of a command key plus at least one other key.

Return Key:

Defines a key sequence consisting of a command key optionally followed by zero or more keys. This key sequence has the following action: terminate the current event and any queued event and return the terminating key sequence to the call processing agent. No default. An application that defines more than one command key sequence, will typically use the same command key for all command key sequences. If more than one command key sequence is defined, then all key sequences must consist of a command key plus at least one other key.

Position Key:

Defines a key with the following action. Stop playing the current announcement and resume playing at the beginning of the first, last, previous, next, or the current segment of the announcement. No default. The actions for the position key are fst, lst, prv, nxt, and cur.

Stop Key:

Defines a key with the following action. Terminate playback of the announcement. No default.

Start Input Keys:

Defines a set of keys that are acceptable as the first digit collected. This set of keys can be specified to interrupt a playing announcement or to not interrupt a playing announcement.

The default key set is 0-9. The default behavior is to interrupt a playing announcement when a Start Input Key is pressed. This behavior can be overidden for the initial prompt only by using the ni (Non-Interruptible Play) parameter. Specification is a list of keys with no separators, e.g. 123456789#.

End Input Key:

Specifies a key that signals the end of digit collection or voice recording. The default end input key is the # key. To specify that no End Input Key be used the parameter is set to the string "null". The default behavior not to return the End Input Key in the digits returned to the call agent. This behavior can be overidden by the Include End Input Key (eik) parameter.

Include End Input Key:

By default the End Input Key is not included in the collected digits returned to the call agent. If this parameter is set to "true" then the End Input Key will be returned with the collected digits returned to the call agent. Default is "false".

Number Of Attempts:

The number of attempts the user needed to enter a valid digit pattern or to make a recording. Defaults to 1. Also used as a return parameter to indicate the number of attempts the user made.

Record Persistent Audio:

If set to true, the recording that is made is persistent instead of temporary. Defaults to false. Valid values are the text strings "true" and "false".

Delete Persistent Audio

Indicates that the specified persistent audio segment is to be deleted. This parameter is carried by the PlayRecord event, although nothing is either played or recorded in this case.

Override Audio:

Indicates that the specified provisioned audio segment is to be overridden with a persistent audio segment to be recorded in the PlayRecord operation that carries this parameter.

Restore Audio:

Indicates that the provisioned audio segment originally associated with the specified segment id is to be restored and that the overriding persistent audio is to be deleted. This parameter is carried by the PlayRecord event, although nothing is either played or recorded in this case.

6. Return Parameters

Each event has an associated set of possible return parameters which are listed in the following tables.

Return Parameters				
Symbol	Definition	pl	pc	pr
vi	voice interrupt			x
ik	interrupting key sequence		x	
ap	amount played		x	x
na	number of attempts		x	x
dc	digits collected		x	
ri	recording id			x
rc	return code	x	x	x

Voice Interrupted:

Set to "true" if the initial prompt of a PlayRecord operation was interrupted by voice. Defaults to "false".

Interrupting Key Sequence:

The key or key sequence that interrupted the initial prompt of a PlayCollect specified using the digit map characters "0" through "9" and "A" through "F" as defined in the DTMF Detection Package in the Megaco protocol document [6].

Amount Played:

The length played of an initial prompt if the prompt was interrupted, in 100 ms units.

Number Of Attempts:

The number of attempts the user needed to enter a valid digit pattern or to make a recording. Defaults to 1. Also used as an input parameter to specify the number of attempts the user will be allowed to enter a valid digit pattern or make a recording.

Digits Collected:

The DTMF digits that were collected during a PlayCollect operation specified using the digit map characters "0" through "9" and "A" through "F" as defined in the DTMF Detection Package in the Megaco protocol document [6].

Recording ID:

A 32 bit binary integer assigned to audio recorded during the Play Record operation.

Return Code:

A return code giving the final status of the operation. Two ranges are defined:

Range	Meaning
100-199	successful completion
300-399	error

The following return codes are define:

Return Code	Meaning
100	Success
300	Unspecified failure
301	Bad audio ID
302	Bad selector type
303	Bad selector value
304	Variable type not supported
305	Variable subtype not supported
306	Invalid variable name
307	Variable value out of range
308	Inconsistent variable specification
309	Alias not found
310	Extra sequence data
311	Missing sequence data
312	Mismatch between play specification and provisioned data
313	Language not set
314	Remove override error
315	Override error
316	Delete audio error
317	Unable to record temporary audio
318	Unable to delete temporary audio
319	Unable to record persistent audio
320	Unable to delete persistent audio
321	Unable to override non-existent segment id
322	Unable to remove override from non-existent segment id
323	Provisioning error
324	Unspecified hardware failure
325	Syntax error
326	No digits
327	No speech
328	Spoke too long
329	Digit pattern not matched
330	Max attempts exceeded

Here are some examples of how the return parameters are used:

The PlayAnnouncement event completed successfully:

```
O: AU/oc(rc=100)
```

The PlayAnnouncement event failed because an alias was not found:

```
O: AU/of(rc=309)
```

The PlayCollect event completed successfully on the user's second attempt when the user entered the digits 04375182:

O: AU/oc(rc=100 na=2 dc=04375182)

The PlayRecord event was successful on the user's first attempt; the id of the recording made by the user is 983:

O: AU/oc(rc=100 na=1 ri=983)

7. Segment Descriptors

Segment descriptors are used with the an, ip, rp, nd, ns, fa, and sa parameters to define the segments that make up an announcement.

Segment Descriptors

Symbol	Definition
32 bit binary number	segment identifier
ts	text to speech
dt	display text
si	silence
to	tone
vb	variable

Segment Identifier:

A 32 bit binary integer identifying a provisioned entity such as a recording, set, sequence, etc.

Text To Speech:

Specifies a text string to be converted to speech.

Display Text:

Specifies a text string to be displayed on a device.

Silence:

Specifies a length of silence to be played in units of 100 milliseconds.

Tone:

Specifies a tone to be played by algorithmic generation. Exact specification of this parameter is tbd. Most tones will likely be recorded, not generated.

Variable:

Specifies a voice variable by type, subtype, and value. Variables are more completely defined in a subsequent section of the document.

8. Variables

The syntax supports two kinds of variables. Embedded embedded variables are variables that have been provisioned as part of a segment. Standalone variables are completely specified in the protocol message.

Typically embedded variables are provisioned along with recorded speech, e.g. "A representative will be with you in approximately 5 minutes. If you would prefer to leave a voice message, press 1 now". where the variable is the number of minutes. This kind of variable is often referred to as an embedded variable.

Variables are specified by the following parameters: type, subtype, and value. Variable types include Date, Money, Number, Time, etc. Subtype is a refinement of type. For example the variable type Money might have an associated range of subtypes such as Dollar, Rupee, Dinar, etc. Not all variables require a subtype, and for these variables the subtype parameter should be set to null.

For embedded variables, the type and subtype must be provisioned. The value may be provisioned. If it is not provisioned it must be specified as part of the variable reference. In a list of segments, an embedded variable value specification applies only to the segment that directly precedes it. If a segment has multiple embedded variables, the values must be given in the order in which the variables are encountered when the segment is played.

Some examples follow below:

A standalone variable:

```
S: pa(an=vb(mny,usd,1153))
```

An embedded variable:

S: pa(an=37<1153>)

Not all variables, such as the date variable shown in the next example, require a subtype. In that case, the subtype is encoded with the value "null":

S: pa(an=vb(dat,null,101598))

In some cases it may be desirable to play an announcement that contains an embedded variable without playing the variable itself. To do this a single "null" is provided for the value:

S: pa(an=37<null>)

Variables Qualifiers				
Symbol	Definition	Type	Subtype	Subtype Of
dat	date	x		
dig	digits	x		
gen	generic		x	dig
ndn	North American DN		x	dig
dur	duration	x		
mth	month	x		
mny	money	x		
num	number	x		
crd	cardinal		x	nm
ord	ordinal		x	nm
sil	silence	x		
str	string	x		
txt	text	x		
dsp	display text		x	txt
spk	text to speech		x	txt
tme	time	x		
t12	twelve hour format		x	tme
t24	twenty four hour format		x	tme
ton	tone	x		
wkd	weekday	x		

Date:

Speaks a date specified as YYYYMMDD (per ISO 8601, International Date and Time Notation [7]). For example "19981015" is spoken as "October fifteenth nineteen ninety eight".

Digits:

Speaks a string of digits one at a time. If the subtype is North American DN, the format of which is NPA-NXX-XXXX, the digits are spoken with appropriate pauses between the NPA and NXX and between the NXX and XXXX. If the subtype is generic, the digits are spoken no pauses.

Duration:

Duration is specified in seconds and is spoken in one or more units of time as appropriate, e.g. "3661" is spoken as "One hour, one minute, and one second".

Money:

Money is specified in the smallest units of a given currency and is spoken in one or more units of currency as appropriate, e.g. "110" in U.S. Dollars would be spoken "one dollar and ten cents". The three letter codes defined in ISO 4217, Currency And Funds Code List [5] are used to specify the currency subtype. A small excerpt from ISO 4217 follows:

Alpha-code	Numeric-code	Currency	Entity
GQE	226	Ekwele	Equatorial Guinea
GRD	300	Drachma	Greece
GTQ	320	Quetzal	Guatemala

Money can be specified in negative or positive units of currency. In the above example "-110" would be spoken as "minus one dollar and ten cents".

Month:

Speaks the specified month, e.g. "10" is spoken as "October". Specification is in MM format with "01" denoting January, "02" denoting February, etc.

Number:

Speaks a number in cardinal form or in ordinal form. For example, "100" is spoken as "one hundred" in cardinal form and "one hundredth" in ordinal form. Cardinal numbers can be specified as negative or positive.

Silence:

Plays a specified period of silence. Specification is in 100 millisecond units.

String:

Speaks each character of a string, e.g. "a34bc" is spoken "A, three, four, b, c". Valid characters are a-z, A-Z, 0-9, #, and *.

Text:

Produces the specified text as speech or displays it on a device.

Time:

Speaks a time in either twelve hour format or twenty four hour format depending on the specified subtype. For example "1700" is spoken as "Five pm" in twelve hour format or as "Seventeen hundred hours" in twenty four hour format. Specification is in HHMM format per ISO 8601, International Data and Time Notation [7].

Tone:

Plays an algorithmically generated tone, specification of which is tbd. Probably most applications will use prerecorded tones.

Weekday:

Speaks the day of the week, e.g. "Monday". Weekdays are specified as single digits, with "1" denoting Sunday, "2" denoting Monday, etc.

9. Selectors

Selector types, except for the pre-defined "Lang" (language) selector, are definable by the user and may be applied to an individual segment within an operation or to all the segments in an operation. For each selector type, the user must also define a range of values that the selector can assume.

For example, if the user defines a selector of type "phase-of-the-moon", he might also define the legal values for that selector to be "new", "half", "full", "harvest", and "blue". For the selector to actually work at runtime, audio associated with each of the selector values must be provisioned.

Although not required, it is suggested that the three letter codes defined in ISO standard 639-2, Code For The Representation Of Names Of Languages [4] be used as values for user defined language selectors. A small excerpt from ISO 639-2 follows:

Code	Language
cze	Czech
cym	Welsh
dan	Danish

Selectors can apply to entire operations or to a segment within an operation. If an operation contains multiple segments, each segment may have its own set of selectors. If selectors for an individual segment and selectors for the entire operation are present, the selector for the individual segment takes precedence for that segment. The selectors for the operation apply to all segments within that operation that do not have individual segment selectors. If a selector set is not specified, provisioned defaults are used.

Selectors are applied to variables only after the variable has been resolved. For instance if a date variable resolved to "October 15th, 1998" the voice with which the variable is spoken could resolve to either male or female if a gender selector had been defined.

10. Aliases

Aliasing of audio segments is supported. The alias to segment id mapping is provisioned and at runtime the alias is resolved to its associated segment id. The syntax for an alias is inclusion of the alias between two forward slashes, e.g.:

```
S: pa(an=/not-in-service/)
```

11. Examples

This section presents a number of examples of how the syntax is used. Note that arguments to an event are separated by a one or more whitespace characters, which can be either an ASCII space character or an ASCII tabulation character.

Play an announcement that consists of a single segment:

```
S: pa(an=39)
```

Play an announcement that consists of multiple segments:

```
S: pa(an=39,40,47)
```

Play an announcement that consists of a recording followed by silence followed by text to speech followed by a standalone voice variable:

```
S: pa(an=39 si(30) ts(hello) vb(my,usd,3999))
```

Play an announcement with an embedded variable. If the first three segments of the previous announcement were provisioned as segment 40, the following would be exactly equivalent to the play in the preceding example:

```
S: pa(an=40<3999>)
```

Play an announcement with two embedded variables:

```
S: pa(an=113<3999,10151998>)
```

Play a prompt and collect a single digit. If need be, play a reprompt, a no digits prompt, and a success or failure announcement. Give the user three attempts to enter a digit:

```
S: pc(ip=21 rp=109 nd=102 fa=81 sa=72 na=3)
```

Play a prompt and collect a single digit. If the user does not enter a digit replay the initial prompt. Give the user three attempts to enter a digit:

```
S: pc(ip=21 na=3)
```

Play a prompt and record voice. If the user does not speak play a no speech prompt. Give the user two attempts to record:

```
S: pr(ip=22 ns=42 na=2)
```

Play an announcement at ninety percent of its original speed and five decibels softer than its original volume. Play the announcement three times with two seconds of silence between plays.

```
S: pa(an=27 sp=90 vl=-5 it=3 iv=20)
```

Give the user two attempts to enter a three digit pattern. Clear the digit buffer before playing the prompt. The user can signal end of input using the # key, which is not returned to the call agent with the collected digits.

S: pc(ip=43 cb=true mn=3 mx=3 na=2)

Give the user three attempts to enter a three digit pattern. If the user enters one digits or two digits on the first or second attempts a reprompt is played. If the user enters no digits on the first or second attempts a no digits reprompt is played. If all three attempts fail, a failure announcement is played. If one of the attempts is successful, a success announcement is played and the collected digits are returned to the call agent. The user can signal end of input using the # key. If the # key terminates a successful input attempt, the collected digits, but not the # key, are returned to the call agent.

S: pc(ip=87 rp=5 nd=409 fa=9 sa=18 mx=3 na=3)

Give the user a single attempt to enter a a 1 to 4 digit pattern, allow 8 seconds for the user to enter the first digit, and allow 6 seconds for the user to enter each subsequent digit. If the subsequent digit timer expires after the user has less than four digits, the digits collected are returned to the call agent. The user can signal end of input using the # key which is not returned to the call agent with the collected digits.

S: pc(ip=4 fdt=80 idt=60 mx=4)

Give the user three chances to enter an 11 digit number that begins with 0 or 1. If the user makes a mistake while entering digits, he can press the * key to discard any digits already collected, replay the prompt, and resume collection.

S: pc(ip=33 mn=11 mx=11 sik=01 rsk=* na=3)

Give the user three chances to enter an 11 digit number that begins with 0 or 1. If the user makes a mistake while entering digits, he can press the key sequence *11 to discard any digits already collected, replay the prompt, and resume collection. If the user enters the key sequence *12 the play collect is terminated along with any queued events, and the terminating key sequence is returned to the call agent for processing.

S: pc(ip=33 mn=11 mx=11 sik=01 rsk=*11 rtk=*12 na=3)

Give the user two chances to make a recording. After playing the prompt, wait 5 seconds for the user to speak, otherwise replay the initial prompt and try again. If the user does speak, wait for seven seconds after speech stops to make sure the user is finished. If the recording is successful, return a reference to the recording to the call agent.

S: pr(ip=6 prt=50 pst=70 na=2)

Play an announcement in the default language:

S: pa(an=5)

Play the same announcement the English. In the first example, the selector applies to the an segment; in the second it applies to the pa operation. For these particular examples, the net effect is the same.

S: pa(an=5[Lang=eng]) or S: pa(an=5)[Lang=eng]

Play an announcement in Danish using a female voice with a Cajun accent.

S: pa(an=6)[Lang=dan,gender=female,accent=cajun]

Play the first part of an announcement in English, the second part in the default language, and the third part in French.

S: pa(an=5[Lang=eng],6,7[Language=fra])

Play an announcement with an embedded variable in English:

S: pa(an=5<101599>)[Lang=eng]

12. Formal Syntax Description

```
AudPkgEvent = PlayAnnouncement / PlayCollect / PlayRecord /
              OperationComplete / OperationFailed / EndSignal

PlayAnnouncement =  [ AdvAudioPkgToken SLASH ]  PlayAnnToken
                    LPAREN PlayAnnParmList RPAREN  [ OpSelectorList ]

PlayCollect  =  [ AdvAudioPkgToken SLASH ]  PlayColToken
                 LPAREN [ PlayColParmList ]  RPAREN  [ OpSelectorList ]

PlayRecord  =  [ AdvAudioPkgToken SLASH ]  PlayRecToken
                LPAREN [ PlayRecParmList ]  RPAREN  [ OpSelectorList ]

OperationComplete  =  [ AdvAudioPkgToken SLASH ]  OpCompleteToken
                LPAREN OpCompleteParmList  RPAREN

OperationFailed  =  [ AdvAudioPkgToken SLASH ]  OpFailedToken
                LPAREN ReturnCodeParm RPAREN
```

```
EndSignal  =  [ AdvAudioPkgToken SLASH ]  EndSignalToken
              LPAREN SignalParm RPAREN

OpSelectorList  =   LSQUARE OpSelector *( COMMA OpSelector ) RSQUARE

OpSelector  =  NAME EQUALS NAME

PlayAnnParmList  =  PlayAnnParm *( WSP PlayAnnParm )

PlayColParmList  =  PlayColParm *( WSP PlayColParm )

PlayRecParmList  =  PlayRecParm *( WSP PlayRecParm )

OpCompleteParmList  =  OpCompleteParm *( WSP OpCompleteParm )

PlayAnnParm  =  ( AnnouncementParm / IterationsParm / IntervalParm /
                  DurationParm / SpeedParm / VolumeParm )

PlayColParm  =  ( InitPromptParm / RepromptParm / NoDigitsParm /
                  FailAnnParm / SuccessAnnParm / NoInterruptParm /
                  SpeedParm / VolumeParm / ClearBufferParm /
                  MaxDigitsParm / MinDigitsParm / DigitPatternParm /
                  FirstDigitParm / InterDigitParm / ExtraDigitParm /
                  RestartKeyParm / ReinputKeyParm / ReturnKeyParm /
                  PosKeyParm / StopKeyParm / StartInputKeyParm /
                  EndInputKeyParm / IncludeEndInputKey /
                  NumAttemptsParm  )

PlayRecParm  =  ( InitPromptParm / RepromptParm / NoSpeechParm /
                  FailAnnParm / SuccessAnnParm / NoInterruptParm /
                  SpeedParm / VolumeParm / ClearBufferParm /
                  PreSpeechParm / PostSpeechParm / RecordLenParm /
                  RestartKeyParm / ReinputKeyParm / ReturnKeyParm /
                  PosKeyParm / StopKeyParm / EndInputKeyParm /
                  RecPersistParm / OverrideAudioParm /
                  RestoreAudioParm /  DeletePersistParm /
                  NumAttemptsParm  )

OpCompleteParm  =  ( VoiceInterruptParm / IntKeySeqParm /
                     NumAttemptsParm / AmtPlayedParm / DigitsColParm /
                     RecordingIdParm / ReturnCodeParm )

AnnouncementParm  =  AnParmToken EQUALS Segmentlist

InitPromptParm  =  IpParmToken EQUALS  Segmentlist

RepromptParm  =  RpParmToken EQUALS Segmentlist
```

```
NoDigitsParm  =  NdParmToken EQUALS Segmentlist

NoSpeechParm  =  NsParmToken EQUALS Segmentlist

FailAnnParm  =  FaParmToken EQUALS Segmentlist

SuccessAnnParm  =  SaParmToken EQUALS Segmentlist

DurationParm  =  DuParmToken EQUALS NUMBER

IterationsParm  =  ItParmToken EQUALS ( NUMBER / "-1" )

IntervalParm  =  IvParmToken EQUALS NUMBER

SpeedParm  =  SpParmToken EQUALS SIGNEDINT

VolumeParm  =  VlParmToken EQUALS SIGNEDINT

NoInterruptParm  =  NiParmToken EQUALS BOOLSTR

ClearBufferParm  =  CbParmToken EQUALS BOOLSTR

MaxDigitsParm  =  MxParmToken EQUALS NUMBER

MinDigitsParm  =  MnParmToken EQUALS NUMBER

DigitPatternParm  =  DpParmToken EQUALS DIGITPATTERN

FirstDigitParm  =  FdtParmToken EQUALS NUMBER

InterDigitParm  =  IdtParmToken EQUALS NUMBER

ExtraDigitParm  =  EdtParmToken EQUALS NUMBER

PreSpeechParm  =  PrtParmToken EQUALS NUMBER

PostSpeechParm  =  PstParmToken EQUALS NUMBER

RecordLenParm  =  RltParmToken EQUALS NUMBER

RestartKeyParm  =  RskParmToken EQUALS CommandKeySequence

ReinputKeyParm  =  RikParmToken EQUALS CommandKeySequence

ReturnKeyParm  =  RtkParmToken EQUALS CommandKeySequence

PosKeyParm  =  PskParmToken EQUALS KeyPadKey COMMA PosKeyAction
```

```
PosKeyAction  =  FirstSegmentToken / LastSegmentToken /
                 PreviousSegmentToken / NextSegmentToken /
                 CurrentSegmentToken

StopKeyParm    =  StkParmToken EQUALS KeyPadKey

StartInputKeyParm  =  SikParmToken EQUALS KeySet

EndInputKeyParm  =  EikParmToken EQUALS KeyPadKey

IncludeEndinputKey  =  IekParmToken EQUALS BOOLSTR

RecPersistParm  =  RpaParmToken EQUALS BOOLSTR

OverrideAudioParm  =  OaParmToken EQUALS SEGID

RestoreAudioParm  =  RaParmToken EQUALS SEGID

DeletePersistParm  =  DpaParmToken EQUALS SEGID

NumAttemptsParm  =  NaParmToken EQUALS NUMBER

VoiceInterruptParm  =  ViParmToken EQUALS BOOLSTR

IntKeySeqParm  =  IkParmToken EQUALS CommandKeySequence

AmtPlayedParm  =  ApParmToken EQUALS NUMBER

DigitsColParm  =  DcParmToken EQUALS KeySequence

RecordingIdParm  =  RiParmToken EQUALS NUMBER

ReturnCodeParm  =  RcParmToken EQUALS  3*3(DIGIT)

KeyPadKey    =  "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9" /
                "*" / "#"

CommandKeySequence  =  1*3(KeyPadKey)

KeySequence = 1*64(KeyPadKey)

KeySet       =   1*11(KeyPadKey)

SignalParm  =  SgParmToken  EQUALS ( PlayAnnToken / PlayColToken /
               PlayRecToken ) RPAREN

Segmentlist  =  SegmentDescriptor  *( COMMA SegmentDescriptor )
```

```
SegmentDescriptor  =  ( ( SegmentId [ EmbedVarList ]
                          [ SegSelectorList ]  ) /
                        ( TextToSpeechSeg [ SegSelectorList ]  ) /
                        ( DisplayTextSeg [ SegSelectorList ]  )  /
                        ( VariableSeg [ SegSelectorList ]  ) /
                          SilenceSeg )

SegmentId  =  ( Segid / Alias )

TextToSpeechSeg  =  TextToSpeechSegToken LPAREN NAME RPAREN

DisplayTextSeg  =  DisplayTextSegToken LPAREN NAME RPAREN

SilenceSeg  =  SilenceSegToken LPAREN NAME RPAREN

VariableSeg  =  VariableSegToken LPAREN FullSpecVar RPAREN

Segid  =  NUMBER

Alias  =  SLASH NAME SLASH

EmbedVarList  =  LANGLE NAME *( COMMA  NAME  ) RANGLE

SegSelectorList  =  LSQUARE SegSelector *( COMMA SegSelector ) RSQUARE

SegSelector  =  NAME EQUALS NAME

FullSpecVar  =  ( DateVariable / DigitsVariable / DurationVariable /
                  MonthVariable / MoneyVariable / NumberVariable /
                  SilenceVariable / StringVariable / TextVariable /
                  TimeVariable / WeekdayVariable )

DateVariable  =   DateVarToken COMMA NullStrToken COMMA Date

Date  =  8*8(DIGIT)

DigitsVariable  =  DigitsVarToken COMMA (NorthAmericanDnToken /
                   GenericDigitsToken) COMMA NUMBER

DurationVariable  =  DurationVarToken COMMA NullStrToken COMMA NUMBER

MoneyVariable  =  MoneyVarToken COMMA 3*3(ALPHA) COMMA OPTSIGNEDINT

MonthVariable  =  MonthVarToken COMMA NullStrToken COMMA Month

Month = "01" / "02" / "03" / "04" / "05" / "06" / "07" / "08" / "09" /
        "10" / "11" / "12"
```

```
NumberVariable  =  (NumberVarToken COMMA CardinalNumberToken COMMA
                    OPTSIGNEDINT) / (NumberVarToken COMMA
                    OrdinalNumberToken COMMA NUMBER)

SilenceVariable  =  SilenceVarToken COMMA  NullStrToken COMMA NUMBER

StringVariable  =  StringVarToken COMMA NullStrToken COMMA *(KeyPadKey)
                   OrdinalNumberToken) COMMA NUMBER

SilenceVariable  =  SilenceVarToken COMMA  NullStrToken COMMA NUMBER

StringVariable  =  StringVarToken COMMA NullStrToken COMMA
                   *(KeyPadKey)

TextVariable  =  TextVarToken COMMA (DisplayTextToken /
                 TextToSpeechToken) COMMA NAME

TimeVariable  =  TimeVarToken COMMA (TwelveHourFormatToken /
                 TwentyFourHourFormatToken) COMMA 4*4(DIGIT)

WeekdayVariable  =  WeekdayVarToken COMMA NullStrToken COMMA NAME

AdvAudioPkgToken       =  "A"
PlayAnnToken           =  "pa"
PlayColToken           =  "pc"
PlayRecToken           =  "pr"
OpCompleteToken        =  "oc"
OpFailedToken          =  "of"
EndSignalToken         =  "es"
TextToSpeechSegToken   =  "ts"
DisplayTextSegToken    =  "dt"
SilenceSegToken        =  "si"
VariableSegToken       =  "vb"
AnParmToken    =  "an"
IpParmToken    =  "ip"
RpParmToken    =  "rp"
NdParmToken    =  "nd"
NsParmToken    =  "ns"
FaParmToken    =  "fa"
SaParmToken    =  "sa"
NiParmToken    =  "ni"
ItParmToken    =  "it"
IvParmToken    =  "iv"
DuParmToken    =  "du"
SpParmToken    =  "sp"
VlParmToken    =  "vl"
CbParmToken    =  "cb"
MxParmToken    =  "mx"
```

```
MnParmToken    =  "mn"
DpParmToken    =  "dp"
FdtParmToken   =  "fdt"
IdtParmToken   =  "idt"
EdtParmToken   =  "edt"
PrtParmToken   =  "prt"
PstParmToken   =  "pst"
RltParmToken   =  "rlt"
RskParmToken   =  "rsk"
RikParmToken   =  "rik"
RtkParmToken   =  "rtk"
PskParmToken   =  "psk"
StkParmToken   =  "stk"
SikParmToken   =  "sik"
EikParmToken   =  "eik"
IekParmToken   =  "iek"
RpaParmToken   =  "rpa"
DpaParmToken   =  "dpa"
OaParmToken    =  "oa"
RaParmToken    =  "ra"
NaParmToken    =  "na"
RidParmToken   =  "rid"
ViParmToken    =  "vi"
IkParmToken    =  "ik"
ApParmToken    =  "ap"
DcParmToken    =  "dc"
RiParmToken    =  "ri"
RcParmToken    =  "rc"
SgParmToken    =  "sg"
DateVarToken       =  "dat"
DigitsVarToken     =  "dig"
DuratioNVarToken   =  "dur"
MoneyVarToken      =  "mny"
MonthVarToken      =  "mth"
NumberVarToken     =  "num"
SilenceVarToken    =  "sil"
StringVarToken     =  "str"
TextVarToken       =  "txt"
TimeVarToken       =  "tme"
WeekdayVarToken    =  "wkd"
GenericDigitsToken         =  "gen"
NorthAmericanDnSToken      =  "ndn"
CardinalNumberToken        =  "crd"
OrdinalNumberToken         =  "ord"
DisplayTextToken           =  "dsp"
TextToSpeechToken          =  "spk"
TwelveHourFormatToken      =  "t12"
TwentyFourHourFormatToken  =  "t24"
```

```
NullStrToken  =  "null"
FirstSegmentToken     =  "fst"
LastSegmentToken      =  "lst"
PreviousSegmentToken  =  "prv"
NextSegmentToken      =  "nxt"
CurrentSegmentToken   =  "cur"
BOOLSTR       =  "true" / "false"
NAMECHAR      =  ALPHA / DIGIT / "_" / "-"
NAME          =  1*64(NAMECHAR)
NUMBER        =  DIGIT *31(DIGIT)
SIGNEDINT     =  ("+" / "-") DIGIT *31(DIGIT)
OPTSIGNEDINT  =  ["+" / "-"] DIGIT *31(DIGIT)
EQUALS        =  "="
COMMA         =  ","
LSQUARE       =  "["
RSQUARE       =  "]"
LANGLE        =  "<"
RANGLE        =  ">"
LPAREN        =  "("
RPAREN        =  ")"
SLASH         =  "/"
WSP           =  SP / HTAB
```

13. References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Arango, M., Dugan, A., Elliott, I., Huitema, C. and S. Pickett, "Media Gateway Control Protocol (MGCP) Version 0.1", RFC 2705, October 1999.

[3] Cromwell, D. and M. Durling, "Requirements For Control Of A Media Services Function", Version 0.0, Work in Progres..

[4] ISO 639-2, "Code For The Representation Of Names Of Languages", 1998.

[5] ISO 4217, "Currency And Funds Code List", 1981.

[6] Cuervo, F., Hill, B., Greene, N., Huitema, C., Rayhan, A., Rosen, B. and J. Segers, "Megaco Protocol", RFC 2885, August 2000.

[7] ISO 8601, "International Date and Time Notation", 1998.

14. Author's Address

David Cromwell
Nortel Networks
Box 13478
35 Davis Drive
Research Triangle Park, NC 27709

Phone: 919-991-8870
EMail: cromwell@nortelnetworks.com

15. Full Copyright Statement

Acknowledgement

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