Suggested Requirements For Control Of An IVR Function
<draft-cromwell-megaco-ivr-req-00.txt>

Status of this Document

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

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

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

To view the entire list of current Internet-Drafts, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net (Northern Europe), ftp.nis.garr.it (Southern Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

Abstract

This document describes the functional requirements for a protocol used by a call processing agent in a packet network to control an interactive voice response function located in the same packet network or at the interface of the packet network and the traditional telephone network. These requirements focus primary on audio, however the protocol could be extended in the future to support other media streams such as video.

The protocol provides the standard audio operations of play audio, collect DTMF, and record speech. It supports direct references to simple audio as well as indirect references to simple and complex
audio. It provides audio variables, interruptibility of audio, 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 dereference complex audio structures. For example, the use could define qualifiers for language, coded, audio format, voice talent, or customer.

The approach used in specifying protocol functionality was to look at several existing protocols currently in use in the telephone network, taking the best concepts from each and attempting to avoid any limitations. The following protocols were examined: ITU CS1-R [2], Nortel Extended CS1-R [3], Bellcore GR-1129 [4], and Bellcore SR-3511 [5]. The protocol described in this document provides at a minimum a superset of the functionality provided by these protocols.
Table of Contents

Notation ...................................................... 5
1. Segments .................................................... 5
   1.1. Introduction ........................................... 5
   1.2. Structural Abstractions ............................... 6
   1.3. Segment Types ......................................... 7
2. Variables .................................................. 8
   2.1. Specification .......................................... 8
   2.2. Inflection ............................................. 9
   2.3. Variable Types ......................................... 10
   2.4. Date ................................................... 11
   2.5. Digits ................................................ 11
   2.6. Duration ............................................... 11
   2.7. Money .................................................. 11
   2.8. Month .................................................. 11
   2.9. Number ................................................ 12
   2.10. Silence ................................................. 12
   2.11. String ................................................ 12
   2.12. Text ................................................... 12
   2.13. Time .................................................. 12
   2.14. Tone .................................................. 12
   2.15. Weekday ............................................... 12
3. Operations ................................................ 12
3.1. Play Operation ............................................ 13
  3.1.1. Announcement ........................................ 13
  3.1.2. Iterations .......................................... 13
  3.1.3. Duration ............................................ 13
  3.1.4. Speed .............................................. 14
  3.1.5. Volume ............................................. 14
  3.1.6. Optional Parameters ................................. 14
  3.1.7. Return Values ..................................... 14
  3.1.8. Examples .......................................... 14
  3.1.9. Return Values ..................................... 14

3.2. Play Collect Operation ................................. 16
  3.2.1. Announcements ..................................... 16
  3.2.2. Speed ............................................. 17
  3.2.3. Volume ........................................... 17
  3.2.4. Interruptibility .................................. 18
  3.2.5. Digit Buffer Control ............................... 18
  3.2.6. Pattern Matching ................................ 18
  3.2.7. Timers ........................................... 18
  3.2.8. Key Definitions ................................... 19
  3.2.9. Number Of Attempts ................................. 20
  3.2.10. Optional Parameters ............................... 21
  3.2.11. Return Values ................................... 21
  3.2.12. Examples ......................................... 21

3.3. Play Record Operation ................................. 23
  3.3.1. Announcements ..................................... 23
1. Segments

1.1. Introduction

A discrete unit of playable speech can be classified as a fragment, a segment, or an announcement. A fragment is the smallest unit and...
typically consists of one or more phonemes, e.g. \\w, the first sound in "welcome." A segment can be either composed of a series of fragments or defined atomically and typically consists of one or more words, e.g. "Welcome" or "Welcome to." An announcement is composed of one or more segments and typically embodies a complete logical expression, e.g. "Welcome to Bell South’s Automated Directory Assistance Service." It is possible for an announcement to be defined as a single segment. In this document "announcement" is a logical concept while "segment" refers to actual audio.

Media operations supported by the protocol should reference announcements. Announcements should be specifiable either as a sequence of one or more segment ids given as a parameter to a media output operation or as a sequence of one or more segment ids provisioned in data and referenced by a single identifier. This identifier can be used as a parameter to a media output operation. Allowing both parameter driven and data driven specification of announcements provides application designers a great deal of flexibility when choosing an application and provisioning model.

A segment id may resolve to a leaf segment containing actual audio or to an intermediate or internal segment which requires further, possibly recursive, resolution to arrive at a leaf segment.

In practice however, the majority of references made by a call processing agent will likely be to a single segment which is a logically complete pre-recorded announcement, e.g. play(27), where segment 27 points to a recording of "Please enter your card number after the tone...<tone>"

1.2. Structural Abstractions

The protocol should support the structural abstractions of set and sequence for storing audio and referencing audio.

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 should not be defined in the protocol itself, but may be defined by the user. Some examples of selector types that a user might define are: language, codec, audio file format, gender, accent, customer, and voice talent.
For example, the user might define a selector of type "language" and then create a set containing English, French, and Russian versions of a particular prompt, say for instance "Please enter your id." At runtime a reference to the set with the selector set to "Russian" would result in the Russian version of the prompt being played.

Recursive definition of both sets and sequences should be allowed, i.e. it should be possible to define a set of sets and a sequence of sequences. In addition, audio structures may also be specified by intermixing sets and sequences, and it should be 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 must not be allowed.

The level of nesting allowed for sequences or sets is implementation dependent. It should be possible to set separate limits on nesting level for each. Real time considerations may dictate practical limitations on nesting level limits.

The protocol should support an arbitrary number of user defined set selectors. For example, language, codec, audio format, customer, voice talent, are all potential set selectors.

1.3. Segment Types

The protocol should support following segment types:

- **RECORDING**: A reference by unique id to a single piece of provisioned audio.
- **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.
- **TO\(NE\)**: 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, language, 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.

Recursive definition of SEQUENCE segments should be allowed. For example sequence A could have as one of its elements sequence B which has as one of its elements sequence C. This feature should be used with caution give the additional complexity it introduces.

Direct or transitive definition of a SEQUENCE segment in terms of itself must not be permitted, e.g. sequence A having as one of its elements sequence B, which has as one of its elements sequence A.

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.

Recursive definition of SET segments should be allowed. For example set A could have as one of its elements set B which has as one of its elements set C. However this feature should be used with caution give the additional complexity it introduces.

Direct or transitive definition of a SET segment in terms of itself must not be permitted, e.g. set A having as one of its elements set B, which has as one of its elements set A.

RESTRICTED SET: The RESTRICTED SET segment type is a SET (as defined above) with the single restriction that the elements of the set must be one of the following segment types: RESTRICTED SET or RECORDING. Use of RESTRICTED SET segments is discussed in the following section.

2. Variables

2.1. Specification

Variables should be specified by the following parameters: type,
subtype, language, 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.

ISO standard 639, Code For The Representation Of Names Of Languages [6], lists the names of many languages and should be used as a starting point in defining the range of available languages. A small excerpt from ISO 639 follows:

<table>
<thead>
<tr>
<th>Code</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs</td>
<td>Czech</td>
</tr>
<tr>
<td>cy</td>
<td>Welsh</td>
</tr>
<tr>
<td>da</td>
<td>Danish</td>
</tr>
</tbody>
</table>

Note that ISO 639 is not a complete list. For example the standard includes Chinese but does not mention the Mandarin or Cantonese dialects.

In some cases it may be desirable to play an announcement with an embedded variable without playing the variable itself. If the value for a variable is NULL, the variable must not be played.

The variable parameters of type, subtype, language, and data are resolved at runtime to a segment id. This segment may be a RECORDING or it may be a RESTRICTED SET segment. The use of RESTRICTED SET segments allows additional levels of indirection beyond those represented by the variable parameters, and could support, for example, variables with any or possibly all of the following: multiple codecs, audio file formats, voice talents, or accents.

2.2. Inflection

Specification of inflection is beyond the scope of this protocol, however a media services function should support rising, flat, and falling inflections as appropriate.
2.3. Variable Types

The protocol should support the following voice variables and should be extensible to support additional variable types. A list of supported variables follows:

<table>
<thead>
<tr>
<th>Type</th>
<th>Subtype</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>none</td>
</tr>
<tr>
<td>DIGITS</td>
<td>GENERIC</td>
</tr>
<tr>
<td></td>
<td>NORTH AMERICAN DN</td>
</tr>
<tr>
<td>DURATION</td>
<td>none</td>
</tr>
<tr>
<td>MONEY</td>
<td>currency_type</td>
</tr>
<tr>
<td>MONTH</td>
<td>none</td>
</tr>
<tr>
<td>NUMBER</td>
<td>CARDINAL</td>
</tr>
<tr>
<td></td>
<td>ORDINAL</td>
</tr>
<tr>
<td>SILENCE</td>
<td>none</td>
</tr>
<tr>
<td>STRING</td>
<td>none</td>
</tr>
<tr>
<td>TEXT</td>
<td>DISPLAY</td>
</tr>
<tr>
<td></td>
<td>SPEECH</td>
</tr>
<tr>
<td>TIME</td>
<td>TWELVEHOUR</td>
</tr>
<tr>
<td></td>
<td>TWENTYFOURHOUR</td>
</tr>
<tr>
<td>TONE</td>
<td>none</td>
</tr>
<tr>
<td>WEEKDAY</td>
<td>none</td>
</tr>
</tbody>
</table>
2.4. Date

Speaks a date specified in MMDDYYYY format. For example "10151998" is spoken as "October fifteenth nineteen ninety eight."

2.5. 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.

2.6. 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."

2.7. 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 list of currency specified in ISO 4217, Currency And Funds Code List [7], should be used as a starting point in defining the currency subtype. A small excerpt from ISO 4217 follows:

<table>
<thead>
<tr>
<th>Alpha-code</th>
<th>Numeric-code</th>
<th>Currency</th>
<th>Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GQE</td>
<td>226</td>
<td>Ekwele</td>
<td>Equatorial Guinea</td>
</tr>
<tr>
<td>GRD</td>
<td>300</td>
<td>Drachma</td>
<td>Greece</td>
</tr>
<tr>
<td>GTQ</td>
<td>320</td>
<td>Quetzal</td>
<td>Guatemala</td>
</tr>
</tbody>
</table>

2.8. Month

Speaks the specified month, e.g. "October."
2.9. 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.

2.10. Silence

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

2.11. String

Speaks each character of a string, e.g. "a34bc" is spoken "A, three, four, b, c."

2.12. Text

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

2.13. Time

Speaks a time (specified in twenty four hour format) in either twelve hour format or twenty four hour format. For example "1700" is spoken as "Five pm" in twelve hour format or as "Seventeen hundred hours" in twenty four hour format.

2.14. Tone

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

2.15. Weekday

Speaks the day of the week, e.g. "Monday."

3. Operations

This section describes the functional requirements for a set of media control operations. Three operations are defined: play, play collect, and play record. Specification of endpoint, port, or channel
on a per operation basis is not a protocol requirement, however it may be required in particular implementations.

3.1. Play Operation

The play operation should play 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 operation is an efficient mechanism for treatments, informational announcements, etc. The play operation should specified as follows:

3.1.1. Announcement

The play operation should play an ordered sequence of one or more segments of the following types: RECORDING, TEXT, SILENCE, TONE, VARIABLE, and SEQUENCE.

3.1.2. Iterations

The protocol should support specification of the maximum number of times an announcement is to be played. It should be possible to specify that an announcement be repeated forever, and it should also be possible to specify an interval of silence (in 100 millisecond units) to be inserted between announcement plays. If the number of iterations is not specified, it should be assumed to be one (i.e. a single play). If the inter-announcement interval is not specified, it should be assumed to be one second.

3.1.3. Duration

The protocol should support specification of the maximum amount of time (in 100 millisecond units) allowed to play and possibly replay an announcement. It should be possible to specify that an announcement be played forever. If duration is specified, it should take precedence over iteration and interval. For example, if a 10 second announcement is to be played 5 times with 2 seconds of silence between plays, the total playing time would be 58 seconds. However, if duration is set to 29, the announcement will only be played for 29 seconds, i.e. the entire announcement will be played twice but the third play will be terminated after 5 seconds.
3.1.4. Speed

The relative playback speed of the announcement should be specifiable as a percentage variation from the normal playback speed. This initial setting should apply to the entire playing of the announcement and should not be changeable. The normal playback speed and the range of change allowed is implementation dependent.

3.1.5. Volume

The relative playback level of an announcement should be specifiable as an percentage variation from the normal playback level. This initial setting should apply to the entire playing of the announcement and should not be changeable. The normal volume and the amount of change allowed is implementation dependent.

3.1.6. Optional Parameters

All parameters to the play operation except the announcement parameter are optional. Certain parameters default to reasonable values. This allows the call agent to specify only the minimum set of parameters it needs in a given situation. If an announcement is not specified an error will be returned to the call agent. The defaults are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>1</td>
</tr>
<tr>
<td>Interval</td>
<td>1 second</td>
</tr>
</tbody>
</table>

3.1.7. Return Values

In addition to a return code that describes the outcome of a play operation, the following information is returned:

- The interrupting key sequence, if any.
- If an announcement was interrupted, the length of the portion of the announcement that was played before the interrupt.
3.1.8. Examples

Assume the following syntax:

\[ \text{play(announcement, iterations, interval, duration, speed, volume)} \] \[ \text{[selectors]} \]

Play a single RECORDING, TEXT, or SEQUENCE segment:

\[ \text{play(segment(5))} \]

Play a single RECORDING, TEXT, or SEQUENCE segment in a female Italian voice.

\[ \text{play(segment(5)) [lang=it, gender=female]} \]

Play a sequence of three segments:

\[ \text{play(segment(5), segment(6), segment(7))} \]

Play three seconds of silence:

\[ \text{play(silence(30))} \]

Play text as speech:

\[ \text{play(speech("hello"))} \]

Display text on a device:

\[ \text{play(display("hello"))} \]

Play "Eleven dollars and fifty three cents" in English:

\[ \text{play(variable(MONEY, USDOLLARS, ENGLISH, 1153))} \]

Specification of a variable without a subtype:

\[ \text{play(variable(DIGITS, NULL, HINDI, 1234))} \]
Play a segment followed by 1 second of silence, followed by "one, two, three, four in Hindi, followed by another segment:

\[
\text{play} (\text{segment}(45), \text{silence}(10), \text{variable}(\text{DIGITS}, \text{NULL}, \text{HINDI}, 1234, \text{segment}(543)))
\]

The same operation as above. The sequence of segment, variable, silence, and segment is defined in data as segment 37:

\[
\text{play} (\text{segment}(37), \text{DIGITS}, \text{NULL}, \text{HINDI}, 1234)
\]

Play an announcement 10% faster than normal speed and 5% softer than normal volume:

\[
\text{play} (\text{segment}(7), \text{speed}(+10), \text{volume}(-5))
\]

Play an announcement three times with two seconds of silence between plays:

\[
\text{play} (\text{segment}(98), \text{iterations}(3), \text{interval}(20))
\]

The same operation as above only the operation is terminated after twenty seconds:

\[
\text{play} (\text{segment}(98), \text{iterations}(3), \text{interval}(20), \text{duration}(200))
\]

3.2. Play Collect Operation

The play collect operation should play a prompt and collect DTMF digits. If no digits are entered or an invalid digit pattern is entered, the user may be reprompted and given another chance to enter the correct digits. The play collect operation should be specified as follows:

3.2.1. Announcements

The play collect operation should optionally play one or more announcements, each consisting of an ordered sequence of one or more segments of the following types: RECORDING, TEXT, SILENCE, TONE, VARIABLE, and SEQUENCE. All play collect announcements are optional and some default to other announcements if they are not specified.

For example if the user fails to enter any digits the no digits
reprompt is played. If the no digits reprompt is undefined then the reprompt is played. If the reprompt is undefined then the initial prompt is played, and if the initial prompt is not defined then no announcement is played.

This concept of cascading defaults allows the level of audio customization to decay gracefully all the way back to a single announcement for all errors and means that applications are not forced to specify any more announcement functionality that they need. The following announcements should be supported for the play collect command. Default relationships are indicated by indentation.

INITIAL PROMPT - If the initial prompt is not specified, digit collection should begin immediately.

REPROMPT - Played after the user has made an error; asks the user to try again. Should default to Initial prompt if not set.

NO DIGITS REPROMPT - Played when the user has not entered any digits. Should default to Reprompt if not set.

FAILURE ANNOUNCEMENT - Played when the all data entry attempts have failed.

SUCCESS ANNOUNCEMENT - Played when the operation has succeeded.

3.2.2. Speed

The relative playback speed of the announcement should be specifiable as a percentage variation from the normal playback speed. This initial setting should apply to the playing of all announcements associated with a particular play collect operation. The normal playback speed and the range of change allowed is implementation dependent.

3.2.3. Volume

The relative playback level of an announcement should be specifiable as a percentage variation from the normal playback level. This initial setting should apply to the playing of all announcements associated with a particular play collect. The normal volume and the amount of change allowed is implementation dependent.
3.2.4. Interruptibility

The play collect operation should support interruptibility by DTMF. A prompt is interruptible if it stops playing when the user presses a DTMF key; if it is non-interruptible it continues to play. Interruptibility should be specifiable in a protocol command on a per segment basis.

3.2.5. Digit Buffer Control

The protocol should support the ability to clear the digit buffer prior to playing the initial prompt. The default should be to not clear the buffer. By default the buffer should always be cleared following the playing of an uninterruptible segment and before playing a reprompt in response to invalid input.

3.2.6. Pattern Matching

The protocol should support specification of the maximum and minimum number of digits to collect. It should support digit pattern matching using extended regular expressions as supported by the Rogue Wave Class Library [8], which supports a subset of the POSIX.2 standard [9] for regular expressions.

3.2.7. Timers

The protocol should support the following event timers for the play collect operation:
FIRST DIGIT - The amount of time allowed for the user to enter the first digit. Specified in units of 100 milliseconds.

INTER DIGIT - The amount of time allowed for the user to enter each subsequent digit. Specified in units of 100 milliseconds.

EXTRA DIGIT - The amount of time to wait for a user to enter a digit once the maximum expected amount of digits have been entered. Specified in units of 100 milliseconds. Typically this timer is used to wait for a terminating key in applications where a specific key has been defined to terminate input.

This timer addresses the "# key ambiguity problem." If the application is expecting 5 digits terminated by the # key, but the digits are valid even if not terminated by the # key, if the digits are sent to the call processing agent as soon as the fifth key is entered, the # key when and if it is received is ambiguous since it could be interpreted as a terminating key for the digits entered previously or as something else.

3.2.8. Key Definitions

The protocol should support the following keys: 0-9, #, *, A, B, C, and D and should provide the ability to specify the semantics of keys received during the play collect operation as defined below. Defined keys are processed in the following order of precedence from highest to lowest: command keys, playcontrol keys, startinput keys, and endinput keys. Any keys not defined should be collected.
COMMAND KEY - A key followed by a sequence of zero or more keys that has one of the following meanings:

RESTART - Discard any digits collected, replay the prompt, and resume collection.

REINPUT - Discard any digits collected and resume collection.

RETURN - Terminate the current operation and any queued operations and return the terminating key sequence to the call processing agent.

PLAYCONTROL KEY - A key that is valid only while an announcement is playing and has one of the following meanings. Play control keys are never collected.

POSITION - Stop playing the current announcement and resume playing at another position within the announcement. A play control key can be defined to resume playing at one of the following positions: the beginning of the first, last, previous, next, or the current segment of the announcement. If the announcement consists of a single segment, the first and previous positions are equivalent to the beginning of the announcement. The last and next positions are equivalent to the end of the announcement.

STOP - Terminate playback of the announcement.

STARTINPUT KEYS - A set of one or more keys that are acceptable as the first digit collected. It should be possible to specify for each key whether interrupts a playing announcement is ignored during a playing announcement.

ENDINPUT KEY - A key that signals the end of user input. It should be possible to specify whether or not this key is included in the collected digits.

The protocol should support specification of the maximum number of times a user may use a restart key to restart the operation or use a reinput key to re-attempt DTMF entry.

3.2.9. Number Of Attempts

The protocol should support specification of the number of times the user can attempt to make a valid entry.
3.2.10. Optional Parameters

All parameters to the play collect operation are optional. Certain parameters default to reasonable values. This allows the call agent to specify only the minimum set of parameters it needs in a given situation. The defaults are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear DTMF</td>
<td>false</td>
</tr>
<tr>
<td>First digit timer</td>
<td>5 seconds</td>
</tr>
<tr>
<td>Interdigit timer</td>
<td>3 seconds</td>
</tr>
<tr>
<td>Start input key</td>
<td>0-9</td>
</tr>
<tr>
<td>End input key</td>
<td>#</td>
</tr>
<tr>
<td>Number of attempts</td>
<td>1</td>
</tr>
</tbody>
</table>

3.2.11. Return Value

In addition to a return code that describes the outcome of a play collect operation, the following information is returned:

The interrupting key sequence, if any.

If an announcement was interrupted, the length of the portion of the announcement that was played before the interrupt.

The number of attempts it took the user to enter a valid sequence of DTMF keys.

The digits that were collected.
3.2.12. Examples

Assume the following syntax:

\[
\text{play\_collect(prompt\_block,timer\_block,key\_block,pattern\_block,speed, volume,cleardigits,attempts)[selectors]}
\]

\[
\text{prompt\_block = (initial\_prompt,reprompt,no\_digits\_reprompt, success\_announcement,failure\_announcement)}
\]

\[
\text{timer\_block = (first\_digit,inter\_digit,extra\_digit)}
\]

\[
\text{key\_block = (command\_block,playcontrol\_block,startinput,endinput)}
\]

\[
\text{pattern\_block = (min\_digits,max\_digits,pattern)}
\]

Clear the digit buffer before initial prompt, play 5% faster than normal speed, 2 percent less than normal volume, and give the user three attempts to enter some valid data:

\[
\text{play\_collect(prompt\_block,timer\_block,key\_block,speed(+5),volume(-2), cleardigits(TRUE),attempts(3))}
\]

Prompt block with only an initial prompt defined:

\[
\text{prompt\_block = (initial\_prompt(87))}
\]

Prompt block with all prompts defined:

\[
\text{prompt\_block = (initial\_prompt(87),reprompt(5), no\_digits\_reprompt(419),failure\_announcement(9), success\_announcement(18))}
\]

Timer block with first_digit timer set to 3 seconds and the inter_digit timer set to 2 seconds:

\[
\text{timer\_block = (first\_digit(3),inter\_digit(2))}
\]

Pattern block specifying collection of 1 to 4 digits:
pattern_block = (min_digits(1),max_digits(4))

Pattern block specifying collection of 2 digits where the first digit is 3, 4, or 5 and the second digit is any digit except 5, 6, or 7.

pattern_block = (min_digits(1),max_digits(2),pattern([3-5][^567]))

Key block specifying a set of digits that are valid as the first digit of input and also specifying that these keys will interrupt the current announcement. Specification of a key to end input. This key is not included in any digits collected.

key_block = (command_block,playcontrol_block,
             startinput(0-9,INTERRUPT),endinput(#,EXCLUDE))

Command_block specifying a restart key sequence and a return key sequence:

command_block = ((*,76,RESTART),(*,83,RETURN))

3.3. Play Record Operation

The play record operation should play a prompt and records user speech. If the user does not speak, the user may be reprompted and given another chance to record. The play record operation is specified as follows:

3.3.1. Announcements

The play record operation should optionally play one or more announcements, each consisting of an ordered sequence of one or more segments of the following types: RECORDING, TEXT, SILENCE, TONE, VARIABLE, and SEQUENCE. All play record announcements are optional and some default to other announcements if they are not specified.

For example if the user does not speak the no speech reprompt is played. If the no speech reprompt is undefined then the reprompt is played. If the reprompt is undefined then the initial prompt is played, and if the initial prompt is not defined then no announcement is played.

This concept of cascading defaults allows the level of audio
customization to decay gracefully all the way back to a single announcement for all errors and means that applications are not forced to specify any more announcement functionality that they need. The following announcements should be supported for the play record command. Default relationships are indicated by indentation.

INITIAL PROMPT - If the initial prompt is not specified, digit collection should begin immediately.

REPROMPT - Played after the user has made an error; asks the user to try again. Should default to Initial prompt if not set.

NO SPEECH REPROMPT - Played when the user has not spoken. Should default to Reprompt if not set.

FAILURE ANNOUNCEMENT - Played when all data entry attempts have failed.

SUCCESS ANNOUNCEMENT - Played when the operation has succeeded.

3.3.2. Speed

The relative playback speed of the announcement should be specifiable as a percentage variation from the normal playback speed. This initial setting should apply to all announcements associated with a particular play record operation. The normal playback speed and the range of change allowed is implementation dependent.

3.3.3. Volume

The relative playback level of an announcement should be specifiable as a percentage variation from the normal playback level. This initial setting should apply to all announcements associated with a particular play record operation. The normal volume and the amount of change allowed is implementation dependent.

3.3.4. Interruptibility

The play record operation should support interruptibility by DTMF. A prompt is interruptible if it stops playing when the user presses a DTMF key; if it is non-interruptible it continues to play. Interruptibility is specifiable in a protocol command on a per segment basis.
3.3.5. Digit Buffer Control

The protocol should support the ability to clear the digit buffer prior to playing the initial prompt. The default should be to not clear the buffer. By default the digit buffer should always be cleared following the playing of an uninterruptible segment, before playing a reprompt in response to invalid input, and before beginning a recording.

3.3.6. Timers

The protocol should support the following event timers for the play record operation:

- **PRE-SPEECH** - The amount of time to wait for the user to initially speak. Specified in units of 100 milliseconds.
- **POST-SPEECH** - 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.
- **TOTAL RECORDING LENGTH** - The maximum allowable length of the recording not including pre or post speech silence. Specified in units of 100 milliseconds.

3.3.7. Key Definitions

The protocol should support the following keys: 0-9, #, *, A, B, C, and D and should provide the ability to specify the semantics of keys received during the play record operation as defined below. Defined keys are processed in the following order of precedence from highest to lowest: command keys, playcontrol keys, and endinput key.
COMMAND KEY - A key followed by a sequence of zero or more keys that has one of the following meanings:

RESTART - Discard any recording in progress, replay the prompt, and resume collection.

REINPUT - Discard any recording in progress and resume collection.

RETURN - Terminate the current operation and any queued operations and return the terminating key sequence to the call processing agent.

PLAYCONTROL KEY - A key that is valid only while an announcement is playing and has one of the following meanings. Play control keys are never collected.

POSITION - Stop playing the current announcement and resume playing at another position within the announcement. A play control key can be defined to resume playing at one of the following positions: the beginning of the first, last, previous, next, or the current segment of the announcement. If the announcement consists of a single segment, the first and previous positions are equivalent to the beginning of the announcement. The last and next positions are equivalent to the end of the announcement.

STOP - Terminate playback of the announcement.

ENDINPUT KEY - A key that signals the end of user input. It should be possible to specify whether or not this key is included in the collected digits.

The protocol should support specification of the maximum number of times a user may use a restart key to restart the operation or use a reinput key to re-attempt recording.

3.3.8. Optional Parameters

All parameters to the play record operation are optional. Certain parameters should default to reasonable values. This allows the call agent to specify only the minimum set of parameters it needs in a given situation. The defaults are:
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear DTMF</td>
<td>false</td>
</tr>
<tr>
<td>Pre speech timer</td>
<td>3 seconds</td>
</tr>
<tr>
<td>Post speech timer</td>
<td>2 seconds</td>
</tr>
<tr>
<td>End input key</td>
<td>#</td>
</tr>
<tr>
<td>Number of attempts</td>
<td>1</td>
</tr>
</tbody>
</table>

3.3.9. Return Values

In addition to a return code that describes the outcome of a play record operation, the following information is returned:

The interrupting key sequence, if any.

If an announcement was interrupted, the length of the portion of the announcement that was played before the interrupt.

The number of attempts it took the user to make a recording.

A reference to any recording that was made.
3.3.10. Examples

Assume the following syntax:

\[
\text{play_record}(\text{prompt\_block}, \text{timer\_block}, \text{key\_block}, \text{speed}, \text{volume}, \\
\hspace{1cm} \text{cleardigits}, \text{attempts})[\text{selectors}]
\]

\[
\text{prompt\_block} = (\text{initial\_prompt}, \text{reprompt}, \text{no\_speech\_reprompt}, \\
\hspace{1cm} \text{success\_announcement}, \text{failure\_announcement})
\]

\[
\text{timer\_block} = (\text{pre\_speech}, \text{post\_speech}, \text{total\_recording\_length})
\]

\[
\text{key\_block} = (\text{command\_block}, \text{playcontrol\_block}, \text{endinput})
\]

Clear digit buffer before initial prompt, play all announcements at 5% faster than normal speed, 2 percent less than normal volume, and give the user only one attempt to make a recording:

\[
\text{play_record}(\text{prompt\_block}, \text{timer\_block}, \text{key\_block}, \text{speed}(+5), \text{volume}(-2), \\
\hspace{1cm} \text{cleardigits}(\text{TRUE}), \text{attempts}(1))
\]

Specify an initial prompt and a reprompt:

\[
\text{prompt\_block} = (\text{initial}(3), \text{reprompt}(45))
\]

Specify prespeech timer of 5 seconds and postspeech timer of 2 seconds:

\[
\text{timer\_block} = (\text{prespeech}(5), \text{postspeech}(2))
\]

4. Other Requirements

This section describes other functional requirements related to media control. These operations do not necessarily directly map to actual commands in a protocol implementation.
4.1. Invoke Application

The protocol should support invocation of a custom application residing on the media services function by application id and accompanying unstructured data block.

4.2. Audio Management

Audio recordings are temporary by default and exist only for the life of the call. The protocol should provide the capability to change at call time the status of a piece of audio from temporary to permanent or from permanent to temporary.

5. Open Issues

The following issues are unresolved:

1. Support for voice recognition.


6. Implementation

Some systems may not be capable of supporting the entire protocol. Implementations of a subset of the protocol should make every attempt to remain logically consistent.
7. References


8. Author’s Address

David Cromwell
Nortel Networks
Box 13010
Research Triangle Park, NC 27709

Phone: (919) 992-1373

e-mail: cromwell@nortel.ca

Michael Durling
Nortel Networks
Box 13010
Research Triangle Park, NC 27709