INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.248.9

(01/2005)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Communication procedures

Gateway control protocol: Advanced media server packages

ITU-T Recommendation H.248.9

ITU-T Recommendation H.248.9

Gateway control protocol: Advanced media server packages

Summary

This Recommendation provides two sets of packages: syntactic and functional. The syntactic packages provide the ability to specify announcements with variable content, with a degree of flexibility constrained only by the provisioning of the MG and MGC. This syntax may in principle be used to specify multimedia announcements, although its application in this Recommendation is to evoke audio content. The functional packages provide advanced control of an Audio Resource Function using the H.248.1 protocol. The packages provide the ability to play recorded announcements with variable content, carry out prompted collection of digits, and carry out prompted collection of recorded audio. An additional package provides the ability to manage recorded media segments on the Media Gateway.

Additional functionality incorporated by H.248.9 Revision 1:

- 6.3.6.11 New variable type "tone" for dynamic audio segment specification.
- 6.4 Set extension of basic syntax: introduction of a new selector for text attributes.
- 6.5.5.1 Variable type "Phrase": introduction of subtypes.
- 9.3.1 Signal PlayCollect: enhanced functionality, new parameters.

NOTE – This Recommendation has been renumbered in 2002. It was formerly known as ITU-T Rec. H.248 Annex M.1.

Source

ITU-T Recommendation H.248.9 was approved on 8 January 2005 by ITU-T Study Group 16 (2005-2008) under the ITU-T Recommendation A.8 procedure.

CONTENTS

ITU-T Recommendation H.248.9

Gateway control protocol: Advanced media server packages

1 Scope

This Recommendation uses the package mechanism to define a parameter syntax to provide a means of referring to provisioned announcements and variable content to be played within them. As indicated in documentation of the packages concerned, this syntax contains optional features, the support of which is indicated by the presence of the additional packages on the termination. The syntax may potentially be used to evoke multimedia content, but, for the most part, that topic is for further study. In addition, this Recommendation adds a series of functional packages to the Megaco/H.248.1 protocol to control an Audio Resource Function which may reside on a Media Gateway or specialized Audio Server.

The announcement specification syntax is described in a series of packages:

- Basic syntax package: Provides the syntax by which to refer to provisioned media segments, with a general capability for extension. See 6.1 for an introduction and 6.2 for detailed definition.
- Voice variables package: An optional extension to the base syntax, which provides stand-alone and embedded variables, with an initial set of voice variable types. See 6.1.4 for an introduction and 6.3 for detailed definition.
- Set syntax package: An optional extension to the base syntax, which 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. See 6.1.5 for an introduction and 6.4 for detailed definition.
- Generic text syntax package: An optional extension to the base syntax, which provides a generic text voice variable type. See 6.1.6 for an introduction and 6.5 for a detailed definition.

The functional packages documented in this Recommendation are as follows:

- Advanced Audio Server (AAS) Base Package: Provides a signal to play an announcement and events to monitor the outcome of the playout request. See clause 8.
- AAS Digit Collection Package: Extends the AAS Base Package by providing a signal and events to coordinate digit collection with the playout of prompting announcements. See clause 9.
- AAS Recording Package: Extends the AAS Base Package by providing a property, signals and events to coordinate the collection of recorded voice with the playout of prompting announcements. See clause 10.
- AAS Segment Management Package allows the MGC to specify an alternative audio segment, which is played in place of a given segment whenever that segment is invoked, until the override is terminated by the MGC. It also allows deletion of persistent segments. Unlike the other packages, this package is defined on a special logical segment control termination and uses only the basic announcement specification syntax. See clause 11.

2 References

2.1 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

– ITU-T Recommendation H.248.1 (2002), *Gateway control protocol: Version 2*, as amended by its Corrigendum 1 (03/2004).

– ITU-T Recommendation Q.1218 (1995), *Interface Recommendation for intelligent network CS-1*.

– IETF RFC 1738 (1994), *Uniform Resource Locators (URL)*.

– IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI): Generic Syntax*.

– IETF RFC 2616 (1999), *Hypertext Transfer Protocol – HTTP/1.1*.

– IETF RFC 3066 (2001), *Tags for the Identification of Languages*.

– ISO 639-1:2002, *Code for the Representation of Languages – Part 1: Alpha-2 code*.

– ISO 639-2:1998, *Code for the Representation of Languages – Part 2: Alpha-3 code*.

– ISO 3166-1:1997, *Codes for the representation of names of countries and their subdivisions – Part 1: Country codes*.

– ISO 3166-2:1998, *Codes for the representation of names of countries and their subdivisions – Part 2: Country subdivision code*.

– ISO 4217:2001, *Codes for the representation of currencies and funds*.

– ISO 8601:2000, *Data elements and interchange formats – Information interchange – Representation of dates and times*.

– ISO/IEC 10646-1:2000 and Amendments, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*.

2.2 Informative references

– IETF RFC 2279 (1998), *UTF-8, a transformation format of ISO 10646*.

– IETF RFC 2326 (1998), *Real Time Streaming Protocol (RTSP)*.

– IETF RFC 2805 (2000), *Media Gateway Control Protocol Architecture and Requirements*.

3 Definitions

This Recommendation defines the following terms:

3.1 audio segment: A separately specifiable unit of audio content. The concept may be generalized to *media segment*, with general multimedia content.

3.2 segment specification: The set of information, which the controller must provide to invoke playout of an audio segment. Potentially segment specifications of the form defined in this Recommendation may also be used to invoke playout of multimedia content, but the details are for further study.

3.3 **voice variable**: A unit of audio content which has one of the types and possibly a subtype as defined in this Recommendation, for which the actual content is given as part of the segment specification. Because the value of a voice variable is specified by text, a voice variable can also be thought of as a text variable if the medium of expression is text.

3.4 **stand-alone variable**: An audio segment whose specification describes a single instance of a voice variable.

3.5 **provisioned segment**: An audio segment which can be retrieved using either a simple identifier or a URI, which must be part of the segment specification. A provisioned segment may include voice variables. The content but not the type/subtype of these variables must also appear within the segment specification.

3.6 **segment set**: A set of alternative forms of expression (e.g., different languages, different speakers) of the same semantic content within an audio segment. The choice of which form of expression to use in a given instance of an audio segment is indicated within the segment specification by giving a value to the selector associated with the set. A given audio segment may be encompassed by multiple sets, with the result that multiple selectors must appear in the segment specification to define a unique instance.

3.7 **selector**: A parameter associated with a set, having a predefined range of values which map to members of the set. Sets, selectors, and the possible ranges of selector values (and default values) are defined by provisioning within the Media Gateway and supporting devices.

3.8 **announcement**: The audible result of playout of a sequence of audio segments. The generation of multimedia announcements is for further study.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations:

AAS	Advanced Audio Server
ABNF	Augmented Backus-Naur Form
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
BR	Brief (type of signal in ITU-T Rec. H.248.1)
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
INAP	Intelligent Network Application Part
MG	Media Gateway
MGC	Media Gateway Controller
OO	On/Off (Type of signal in ITU-T Rec. H.248.1)
RTSP	Real Time Streaming Protocol
TO	Timeout (type of signal in ITU-T Rec. H.248.1)
UCS	Universal Character Set
URI	Universal Resource Identifier
URL	Universal Resource Locator
UTF	UCS Transformation Format

5 Design philosophy

The syntax packages in this Recommendation are a formal device whereby the MGC can determine the level of capability of the MG to process particular constructs within the announcement specification syntax on a given termination. The MGC acquires this information by audit. The presence of a given syntactic package indicates the ability to process the syntax described in the procedural section of that package. The syntax packages have no content other than these procedures.

The functional packages in this Recommendation provide significant capabilities most of which are controlled via protocol parameters. Most parameters are optional, and generally can be omitted in favour of their default values. An audio application that invokes references to complex provisioned audio structures can specify audio events using a minimum of syntax by taking advantage of parameter optionality and parameter defaults.

The operations covered in this Recommendation are invoked as signals on a termination associated with the user (except for the announcement override operations, which are signals invoked on a special logical segment control termination). That basic mechanism has aspects, which require care when using the play-and-collect-digits and play-and-record packages. The main problem is to avoid unintended interruption of these operations due to the recognition of events on the termination. The MGC must ensure that the KeepActive flag is set on events it enables which are not intended to stop these operations.

The usual rules for Signals Descriptor replacement apply to the signals described by the functional packages in this Recommendation. That is, if the Signals Descriptor, which invoked a given operation, is replaced, the operation will continue without interruption only if it is identically invoked in the new Signals Descriptor, with the signal KeepActive flag set.

The play-and-collect-digits operation uses the H.248.1 digit map descriptor to indicate the expected pattern of digits to be collected. However, the interaction required with the user in the case of failure to collect the expected pattern on the first attempt precludes use of the full digit map mechanism built into H.248.1. Instead, it has been necessary to specify modified behaviour and to provide the results in an operation completion event rather than a digit map completion event.

6 Announcement specification syntax

6.1 Syntactical concepts: audio segments, variables, and embedded variables

All packages in this Recommendation rely on the use of a special parameter syntax to describe the announcements to be played out. This syntax allows announcements to be described as a series of audio segments, each of which has either been provisioned at some physical location or is dynamically specified by the announcement description itself (in the form of a stand-alone voice variable).

The Base Announcement Syntax Package supports both simple and complex audio structures. A simple audio structure might be a single announcement such as "Welcome to the 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.

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 segment specification syntax supports both approaches.

The syntax described in this Recommendation has three components: the basic syntax which must be supported by all implementations of the packages in this Recommendation, the syntax supporting the use of "sets" to qualify announcement playout, and a syntax supporting arbitrary text variables. Capabilities beyond the base syntax are optional; their support is indicated by the presence of the corresponding packages on the termination on which playout is invoked.

6.1.1 Provisioned audio segments

It is possible that a single reference to a provisioned audio segment actually invokes a complex audio structure, including variables whose values are to be specified at the time of invocation. The syntax allows the MGC to specify the values of such embedded variables. With this exception, the difference between simple and complex provisioned audio segments is invisible to the MGC and irrelevant to the protocol.

The syntax uses URIs (Universal Resource Identifier) to designate provisioned segments, with the result that they can be physically located either on the MG or on some other device, without affecting the message flows between the MGC and MG. Every provisioned segment is assigned a unique URI which among other things can be a hierarchical name, or a simple name or number.

6.1.2 Dynamically specified audio segments

A dynamically specified audio segment is one specified by a stand-alone voice variable. See 6.1.4 for more information on variables.

6.1.3 Segment identifiers

Provisioned segments and segments recorded at run time are identified by URIs as defined in IETF RFC 2396.

A URI can be a simple name or it can be a URL. Three URL schemes are allowed: the file: scheme, the ftp: scheme, and the http: scheme. The file: scheme is used for audio local to the Audio Server. The ftp: scheme is used for audio remote to the Audio Server. The http: scheme can be used for audio local to the Audio Server using the http://localhost convention or for audio remote to the Audio Server. All audio references that require parameters encoded in the URL must use the http: scheme. The following examples show some of the possibilities. More examples are shown in 6.6.

NOTE – For playout of more general media over IP transport the rtsp: scheme should also be considered. Multimedia announcement specifications are for further study.

Reference to local audio (simple name):	12354
Reference to local audio (flat file):	file://welcome
Reference to local audio:	file://audio/xyztel/welcome
Reference to remote audio:	http://audio/xyztel/welcome

6.1.4 Variables

A voice variable represents a single semantic concept (such as date or number) and dynamically produces the appropriate speech based on information supplied at run time. For example, if an application needs to play a date, rather than telling the AudioServer to play each individual component of the date (e.g., "March" "twenty" "second" "nineteen" "ninety" "nine"), the MGC can specify a voice variable of type Date with value "19990322". The Audio Server then assembles and plays the component audio needed to speak the date.

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 must be set to null.

As described above, the AAS announcement syntax supports two kinds of variables: stand-alone and embedded. Stand-alone variables are variables that are not part of a provisioned audio segment. Their type, subtype, and value must be completely specified by the MGC. This specification constitutes a dynamically specified audio segment as described above.

Embedded variables are variables that have been provisioned as part of a provisioned audio segment. At run time the MGC references the segment and specifies a value for each variable embedded in 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.

6.1.4.1 Example of use of variables in a sequence

In the following example, the sequence to be played speaks the following: "Today's date is <weekday> <date>." This sequence is made up of three segments: a simple audio segment, a variable of type Weekday, and another variable of type Date. The sequence can be implemented in two ways: as a sequence explicitly specified by the MGC, or as a single provisioned audio segment with two embedded variables. These two approaches are illustrated by Figures 1 and 2, respectively.

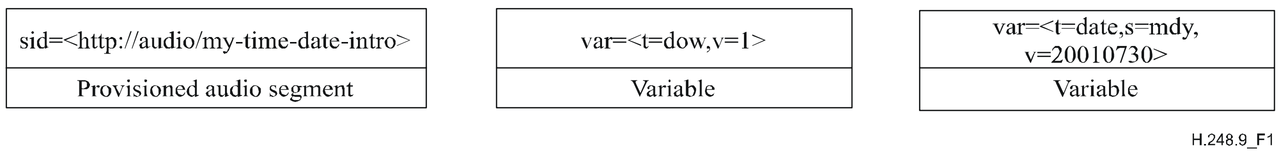
sid=<http://audio/my-time-date-intro>
Provisioned audio segment
var=<t=dow,v=1>
Variable
var=<t=date,s=mdy, v=20010730>
Variable
H.248.9_F1

Figure 1/H.248.9 – Explicit sequence with three audio segments

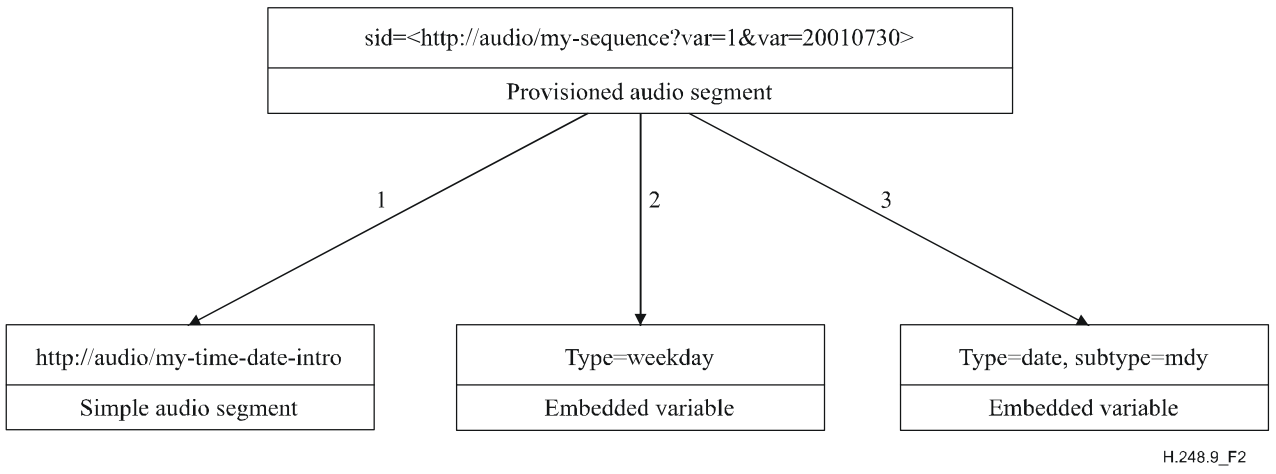
sid=<http://audio/my-sequence?var=1&var=20010730>
Provisioned audio segment
1
2
3
http://audio/my-time-date-intro
Simple audio segment
Type=weekday
Embedded variable
Type=date, subtype=mdy
Embedded variable
H.248.9_F2

Figure 2/H.248.9 – Provisioned audio segment with two embedded variables

In both cases, the provisioner has installed a simple audio segment designated by http://audio/my-time-date-intro. In the first case, this segment is visible to the MGC. In the second case, the MGC only knows about the provisioned segment http://audio/my-sequence, which contains an embedded weekday variable and an embedded date variable in that order. The fact that http://audio/my-sequence itself references http://audio/my-time-date-intro is known only at the device to which http://audio/my-sequence resolves.

6.1.5 Segment sets

Sets are an advanced, optional feature of the announcement specification syntax. A set is a provisioned collection of alternative audio segments and an associated selector. Each set is assigned a unique URI. At run time the value of the selector is used to determine which element of the set is played. Within an announcement specification, a set appears as a single provisioned audio segment with its selector value(s).

Individual selector types are not defined in the syntax (except for the pre-defined language selector) and are instead defined by the provisioner. A provisioner could, for example, define one or more of the following selector types: language, accent, gender, accent, customer, and/or day of the week. For each selector type, the provisioner must define a range of valid values. The provisioner may also choose to define a default value. If a selector value is not supplied at run time, the default value is used.

Multidimensional sets are permitted. These support a vector of selector types. A value must be specified for each selector type in order to resolve to a specific instance of the audio segment concerned.

A set can contain embedded variables. The type and order of these must be the same for every member of the set. The playout of an embedded variable must be consistent with the value of the selector used to invoke the audio segment in which it is embedded. Thus, for example, invocation of a provisioned audio segment associated with a language selector and containing an embedded date variable must result in a playout of the date value provided in the invocation in the language indicated by the selector value. As with other segments that can contain variables, if a set has multiple embedded variables, the variable values must be specified in the order in which the variables are encountered when the segment is played. Sets in which variables must be played in different orders depending on selector value are not supported.

6.1.5.1 Set example

Figure 3 has an example of a set. To support an application which plays a particular piece of audio in either Arabic, Welsh, or Tibetan, a provisioner could define a set with the predefined selector, "lang", and define three of the possible values for that selector, "ar", "cy", and "bo". The provisioner would provision three audio segments, one in each language, and would associate the Arabic segment with the "ar" selector value, etc. The provisioner also could define a default value of the selector when no selector value is supplied, "ar" for instance. The entire set would be assigned a unique URI, which would be the only URI visible to the MGC.

At run time, a reference to the set with the selector set to "cy" would result in the Welsh version of the prompt being played. A reference to the set with no selector would result in the Arabic version of the prompt being played since Arabic has been set as the default selector value.

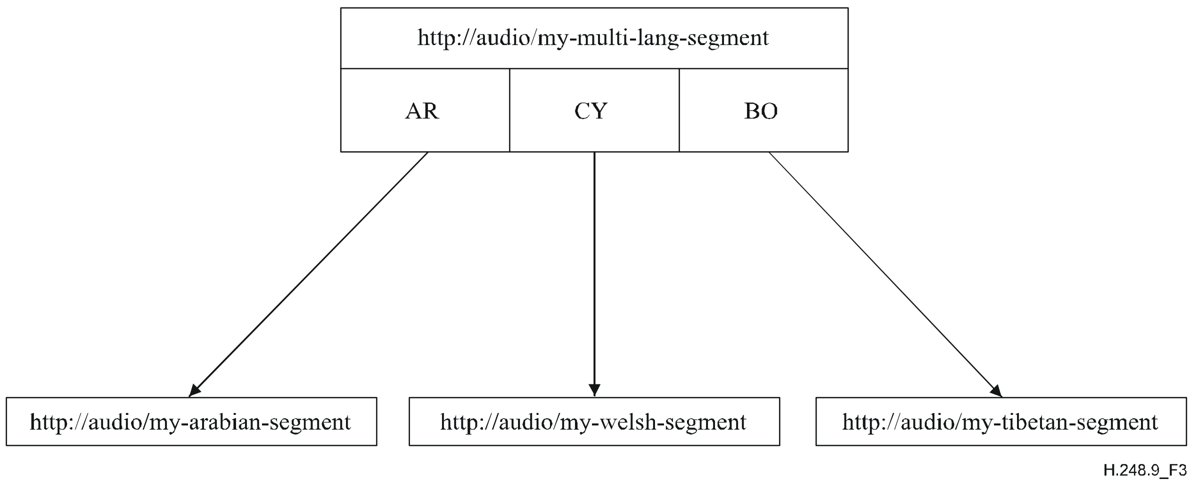
http://audio/my-multi-lang-segment
AR
CY
BO
http://audio/my-arabian-segment
http://audio/my-welsh-segment
http://audio/my-tibetan-segment
H.248.9_F3

Figure 3/H.248.9 – Set example

6.1.5.2 Example of set with embedded variable

In this example, the provisioner has provisioned three sequences, one in Arabic, one in Welsh, and one in Tibetan, each consisting of a simple audio segment followed by a date variable. This is illustrated in Figure 4. The provisioner has assembled these into a set consisting of the three sequences with language as the set selector. Again, the only part of this visible to the MGC is the URI referring to the entire set, the language selector, and the embedded date variable.

At run time a reference to the set with the selector set to "ar" and a variable value of "20001015" would result in the following being played in Arabic: "Today's date is October 15th, 2000."

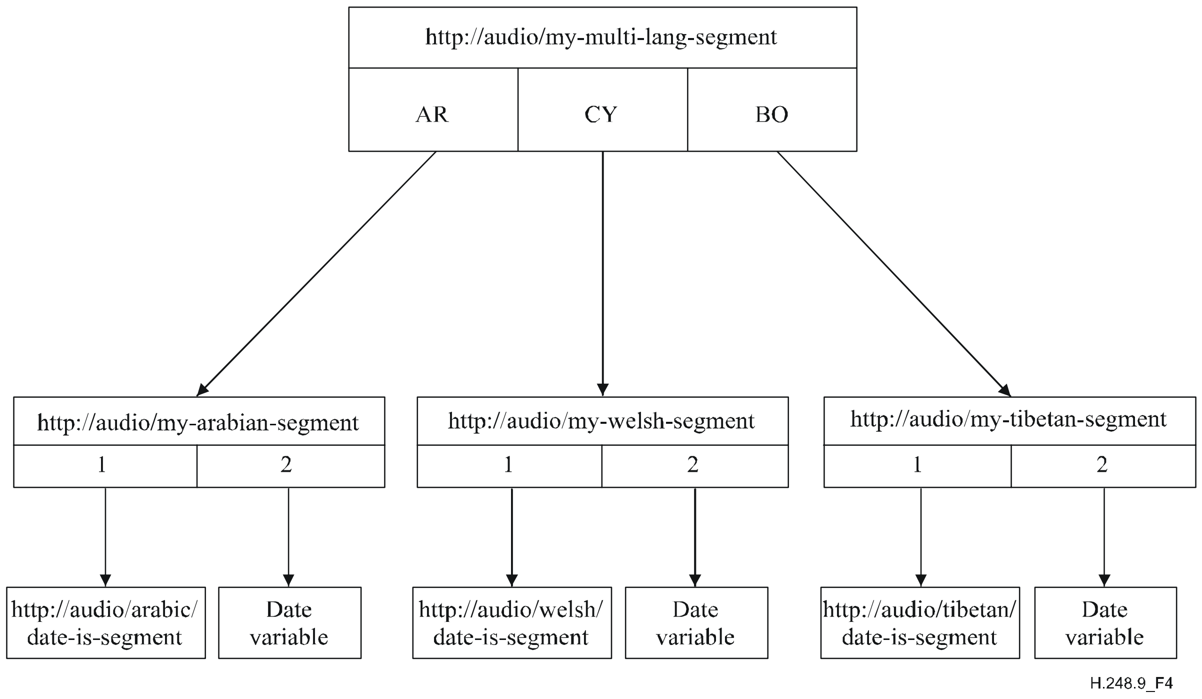
http://audio/my-multi-lang-segment
AR
CY
BO
http://audio/my-arabian-segment
1
2
http://audio/my-welsh-segment
1
2
http://audio/my-tibetan-segment
1
2
http://audio/arabic/ date-is-segment
Date variable
http://audio/welsh/ date-is-segment
Date variable
http://audio/tibetan/ date-is-segment
Date variable
H.248.9_F4

Figure 4/H.248.9 – Example of set with embedded variable

6.1.6 Generic text variables

The syntax provides an optional capability to speak an arbitrary variable phrase. The phrase is represented in the segment specification using a UTF-8 encoding (IETF RFC 2279) of the default writing system provisioned for the MG. Depending on the capabilities of the MG, the language in which it is spoken may be provisioned or may be indicated by use of the language selector. The capability is provided in the form of an additional voice variable type.

6.2 Basic announcement syntax package

Package name:	Basic Announcement Syntax
Package ID:	bannsyx (0x0047)
Description:	This package exists only to indicate that the MG is capable of processing the syntax described herein. An MGC learns that the capability is supported by auditing the packages supported by the termination on which playout is to be performed and verifying that this package is listed.

The syntax defined in this clause is used to designate announcements to be played out by the various Advanced Audio Server signals defined in this Recommendation. This syntax may also be used to designate multimedia content, although extensions (such as additional URL types) for that purpose may be desirable.

Version:	1
Extends:	none.

6.2.1 Properties

None.

6.2.2 Events

None.

6.2.3 Signals

None.

6.2.4 Statistics

None.

6.2.5 Procedures

6.2.5.1 General structure

An announcement specification consists of one or more segment specifications. Each segment specification describes either a provisioned audio segment (with possible embedded variables) or a stand-alone voice variable.

NOTE – While the general structure just described is easily generalizable to multimedia content, the use of variables is one of the issues requiring further study for that case. To make the basic syntax immediately applicable to multimedia content, voice variables are described in a separate package.

6.2.5.1.1 ASN.1 encoding

In the ASN.1 encoding, the general signal parameter syntax is ultimately constrained by the Value production of Annex A/H.248.1. Parameter values are double-wrapped with an inner BER encoding applied first to aid interpretation of the parameter, followed by an outer BER encoding as an OCTET STRING. The general structure of the basic AAS announcement specification syntax for purposes of the inner encoding is expressed as follows:

```
AnnouncementSpec ::= IA5String
```

The details of the string structure are as specified in the remainder of clause 6: they are equally applicable to text and ASN.1 encoding.

6.2.5.1.2 Text encoding

In the text encoding, the detailed signal parameter syntax is ultimately constrained by the VALUE production of Annex B/H.248.1. The ABNF description of the gross structure of an announcement specification is as follows:

```
announcementSpec = DQUOTE segSpec *( COMMA segSpec ) DQUOTE
  ; DQUOTE and COMMA are as defined in Annex B/H.248.1.

segSpec = keyword "=" "<" spec ">" ; angle brackets as delimiters

keyword = "sid"          ; provisioned segment identifier
        / "var"          ; standalone variable

spec    = provSegSpec    ; provisioned segment identifier
        / varSegSpec     ; standalone variable

varSegSpec = varSpec     ; additional general level to facilitate selector
                         ; extension
```

The quotedString form of VALUE is required for announcementSpec because a segSpec can contain restricted characters (e.g., =, <, > as shown above), and because successive segSpecs are comma-separated. However, the VALUE production requires escapes for the following:

- all control characters (%x00-%x1F and %x7F) except TAB (%x09) ;
- the DQUOTE character (%x22).

Outside of URIs, the issue of escaping only arises in connection with general character sequences, which are possible with the Chars and Phrase variable types. (See 6.5 for the latter.) This specification represents general UTF-8 characters in the U+xxxx form to avoid the need to escape the individual byte values.

Escaping within URIs must be performed as described in IETF RFC 2396. Escaping within stand-alone voice variable specifications uses the same mechanism as IETF RFC 2396, but applies only to the characters listed above, the percent sign "%" (which is used as an escape character), and the closing angle bracket ">" (which terminates a variable value). (This is currently a non-issue, since neither "%" as a non-escape character nor ">" will be found within any variable value defined in this Recommendation.)

Keywords in the text encoded syntax are case-insensitive. Case sensitivity within URIs is defined by the applicable standards. Variable values are case-sensitive only where this is explicitly specified.

6.2.5.2 Provisioned segment specifications

A provisioned segment specification consists of either a simple name or a URI formed under the rules of IETF RFC 2396. The syntax of a simple name is slightly broader than the NAME construct in Annex B/H.248.1, because it is not required to begin with an alphabetic character. This Recommendation supports three URI schemes:

- the file: scheme, used for provisioned segments local to the MG;
- the ftp: scheme, used for segments on a device remote from the MG;
- the http: scheme, used for segments located either locally or remotely to the MG. Segments located locally to the MG must use "localhost" as the <host> part of the URI.

The MGC must use the http: scheme if the provisioned audio segment contains embedded variables. It must also use the http: scheme if the segment supports selectors (see 6.4). This restriction is necessary because the announcement specification syntax uses the http: scheme query part to carry embedded variable (and selector) values.

In accordance with IETF RFC 2396, the following characters must be escaped within all URIs:

- reserved characters within the individual URI schemes. IETF RFC 1738 is the most recent description of the file:, ftp:, and http: schemes. According to this RFC, "/" is reserved for separating components of a path hierarchy, ";" is reserved within the ftp: and http: schemes, and "?" is reserved in the http: scheme;
- the space character;
- characters used as delimiters or for escaping: "<", ">", "#", "%", and <">;
- characters subject to unwanted transformations or subject to misinterpretation: "{", "}", "|", "\", "^", "[", "]", and "`".

6.2.5.2.1 Text encoding

This clause provides a detailed description of the provSegSpec production which is referred to in 6.2.5.1.2.

```
provSegSpec = simple / ftpurl / httpurl / fileurl

simple = 1* ( ALPHA / DIGIT / "_" )
  ; ALPHA and DIGIT as defined in Annex B/H.248.1

fileurl = "file://" host path
  ; See RFC 1738 for further details. "file://" is case-sensitive.

ftpurl = "ftp://" [user [":" password ] "@" ] host [":" port ]
       [ "/" *(cwd "/") name [";type=" type] ]
     ; See RFC 1738 for further details. "ftp://" is case-sensitive.

httpurl = "http://" host [ ":" port ] [ abs_path [ "?" query ]]
     ; Omit "?" if query is empty.
     ; See RFCs 1738 and 2616 for further details. "http://" is
       case-sensitive.
```

Where the httpurl form is used, the query part must be present if the provisioned audio segment contains embedded variables (or supports selectors, see 6.4). The general form of the query part is as follows:

```
query = category "=" catVal  *( "&" category "=" catVal)
category = 1*ALPHA
  ; Case-insensitive
```

In addition to the character escaping rules already described, it is required that the "&" character be escaped (replaced by "%26") if present within a catVal.

6.3 Voice variable syntax package

Package Name:	Voice Variable Syntax
PackageID:	vvsyx (0x0048)
Description:	This package exists only to indicate that the MG is capable of processing the syntax described herein. An MGC learns that the capability is supported by auditing the packages supported by the termination on which playout is to be performed and verifying that this package is listed.

The syntax defined in this clause is used to designate voice variables, either as embedded variables within announcement segments, or as stand-alone variables. Because the variable values are specified as text, voice variables may also be used as text variables when the announcements are expressed as text.

Version:	2
Extends:	bannsyx version 1.

6.3.1 Properties

None.

6.3.2 Events

None.

6.3.3 Signals

None.

6.3.4 Statistics

None.

6.3.5 Procedures

6.3.5.1 Embedded variables

When embedded variables are present, their values are provided as successive ampersand-separated components of the query part defined in 6.2.5.2.1. One value is provided per embedded variable, in the order of embedding. Formally, the syntax of an embedded variable is represented by the following extension to the syntax of 6.2.5.2.1:

```
category =/ "var"
catVal =/ varVal
varVal = genval / default / empty

genval = 1* (SafeChar / RestChar / WSP)
  ; SafeChar, RestChar, and WSP as defined in Annex B/H.248.1.
  ; Escaping required as indicated in this clause and 6.2.5.2.
  ; The text encoding is given by the portion of the production for the
  ; applicable type in 6.3.6 (and 6.5) which follows
  ; the "v="tag.

default = "-"
  ; Single character "-" followed by "&" or ">" delimitor indicates
  ; that the executing host should use the provisioned default value,
  ; if any, of the embedded variable.

empty = ""
  ; Empty string (i.e. delimiter immediately following "var=").
  ; Indicates that embedded variable must not be
  ; played out.
```

It is an error for the MGC to request playout of a default value if none is provisioned. Error code 607 is applicable, whether the error is reported in the transaction response or by means of the Audio Operation Failure event defined in the AAS Base package.

6.3.6 Dynamic audio segment specifications (Stand-alone voice variables)

A dynamic audio specification consists of a type, a possible subtype, and value for a single variable. The possible range of subtypes varies with the variable type. The basic syntax supports the types and subtypes listed below.

The text encoding descriptions within this clause extend the varSpec production, which is referred to in 6.2.5.1.2 to include voice variables.

A stand-alone variable specification includes the type, possible subtype, and value. Each of these components is introduced by a tag: "t=", "s=", and "v=" respectively. Successive components are separated by commas. The value is the set of characters following "v=" and preceding the closing ">" of the segSpec. The escaping rules of 6.2.5.1.2 must be applied to variable values as required.

```
     varSpec =/ vvarSpec   ; Voice variable specification
```

The varSpec production is extended in 6.4.5.3.2 to include selectors. The vvarSpec production is extended in the following subclauses to include detailed specifications by variable type.

6.3.6.1 Variable type: Time

Definition: Speaks a time of day.

Subtypes: The subtypes associated with the Time variable specify the format in which the time is spoken (12 hour format and 24 hour format). In many languages, however, it only makes sense to speak the elements of the time in one format. If a language provides more than one way to speak time, subtype can be used to override the default alternative. If a language provides only a single way to speak time, subtype can be omitted; if subtype is specified in this case it will be ignored.

Value: A string of four digits giving a time specified as HHMM (per ISO 8601), in twenty-four hour format.

Example: "1700" is spoken as "Five pm" in twelve hour format or as "Seventeen hundred hours" in twenty-four hour format.

Text encoding:

```
     vvarSpec =/ todSpec

     ; Time of day
     todSpec = "t=tod" [ ",s=" ( "t12" / "t24" ) ]   ",v=" 4DIGIT
       ; Subtype selects 12- or 24-hour format.
       ; Value is HHMM per ISO 8601.
```

6.3.6.2 Variable type: Weekday

Definition: Speaks the name of a specified day of the week.

Subtypes: Not applicable.

Value: A single digit character, beginning with "1" denoting Sunday and ending with "7" denoting "Saturday".

Example: "2" is spoken as "Monday".

Text encoding:

```
vvarSpec =/ dowSpec

 ; Day of week (weekday)
dowSpec =  "t=dow" ",v=" %d1-7
     ; "1" is Sunday ... "7" is Saturday
```

6.3.6.3 Variable type: Date

Definition: A date made up of three components: day of month, name of month, and year.

Subtypes: The subtypes associated with the Date variable specify the order in which the elements of the date (day, month, and year) are spoken. In many languages, however, it only makes sense to speak the elements of the date in one particular order. If a language provides more than one way to speak date, subtype can be used to override the default alternative. If a language provides only a single way to speak-date, subtype can be omitted; if subtype is specified in this case it will be ignored.

Value: The value is a string of eight digits specifying a date in the form YYYYMMDD (per ISO 8601).

Example: The value "20001015" could be spoken as "October Fifteenth Two Thousand" or as "Fifteen October Two Thousand" depending on the subtype.

Text encoding:

```
vvarSpec = / dateSpec

; Date
dateSpec = "t=date" [ ",s=" dateorder ]   ",v=" 8DIGIT
  ; Subtype determines order in which components are spoken.
  ; Value is in form YYYYMMDD per ISO 8601.

; Order subtype is separated to make it extensible if desired.
dateorder = "mdy"              ; month-day-year
          / "dmy"              ; day-month-year
```

6.3.6.4 Variable type: Month

Definition: Speaks the name of the specified month.

Subtypes: Not applicable.

Value: A two-digit string of digits in MM format with "01" denoting January, "02" denoting February, etc.

Example: "10" is spoken as "October".

Text encoding:

```
vvarSpec = / monthSpec

; Month
monthSpec = "t=month" ",v=" 2DIGIT
  ; "01" is January ..."12" is December
```

6.3.6.5 Variable type: Duration

Definition: A period of time spoken in one or more units of time as appropriate.

Subtypes: Not applicable.

Value: An integer giving a number of seconds.

Example: Value "3661" is spoken as "One hour, one minute, and one second."

Text encoding:

```
vvarSpec = / durSpec

; Duration
durSpec = "t=dur" ",v=" 1*DIGIT
```

6.3.6.6 Variable type: Digits

Definition: A sequence of digits which are spoken one at a time.

Subtypes: Not applicable.

Value: A string of digits of arbitrary length, given in the order they are to be spoken.

Example: Type Digits, value "61360961" spoken as "six one three six zero nine six one".

Text encoding:

```
vvarSpec = / digitSpec
; Sequence of digits
digitSpec = "t=digits"  ",v=" 1*DIGIT
```

6.3.6.7 Variable type: Chars

Definition: Speaks a specified sequence composed of upper and lower case alphabetic characters (if case is applicable to the writing system involved), digits, and the special characters # and *. The alphabetic characters are case-sensitive (again, if applicable).

Subtypes: Not applicable.

Value: Valid characters in the ASCII character set are a-z, A-Z, 0-9, #, and *. Note that it is necessary to escape the character # when present in the value of an embedded variable. Restrictions of characters in other writing systems require further study, but should reflect the intention that this variable type be used to spell out dialling prompts, telephone numbers, or names and addresses.

Text encoding:

```
vvarSpec = / charSpec

; Sequence of characters
charSpec = "t=chars" ",v="
          ( 1*(  LOWALPHA / UPALPHA / DIGIT / ( "#" / "%23" ) / "*"  )
            ; ASCII string, restricted to (a-z, A-Z, 0-9, #, *)
            ; Note - need to escape "#" within the value of an embedded
              variable
          / ( "U+" 2*12HEX) *( "." 2*12HEX ) ) )
            ; General UTF-8 string as a sequence of dot-separated
              hex-encoded
            ; values introduced by "U+", representing 1 to 6 octets.
; LOWALPHA, UPALPHA, DIGIT, HEX as defined in RFC 2396
```

6.3.6.8 Variable type: Money

Definition: An amount of money of a given currency, spoken in mixed units of that currency as appropriate, and as either a positive or a negative quantity as indicated by the sign of the value.

Subtypes: The different currency types as specified by ISO 4217. A small excerpt from ISO 4217 follows:

Alpha-code	Numeric-code	Currency	Entity
VEB	862	Bolivar	Venezuela
VND	704	Dong	Viet Nam
USD	840	US Dollar	Virgin Islands (British)

Value: An optionally signed integer giving a quantity of money specified in the smallest units of a given currency.

Example: "110" in U.S. Dollars would be spoken "one dollar and ten cents."

Text encoding:

```
vvarSpec = / moneySpec

; Amount of money (positive or negative)
moneySpec = "t=money" [",s=" 3ALPHA] ",v=" [ "-" ] 1*DIGIT
  ; Subtype is ISO 4217 alpha-code
```

6.3.6.9 Variable type: Integer

Definition: Speaks an integer.

Subtypes: Control whether the number is spoken as a cardinal or ordinal value.

Value: An optionally signed integer. Negative integers are allowed only with the cardinal subtype.

Example: "100" is spoken as "one hundred" in cardinal form and "one hundredth" in ordinal form.

Text encoding:

```
vvarSpec = / intSpec

; Integer (ordinal or positive or negative cardinal)
intSpec = "t=int" [ ",s=" ( "card" / "ord" ) ] ",v=" [ "-" ] 1*DIGIT
  ; Negative values allowed only for cardinal numbers.
```

6.3.6.10 Variable type: Silence

Definition: A period of silence of a specified duration.

Subtypes: Not applicable.

Value: An unsigned integer giving the duration of the period of silence in 100 millisecond units.

Example: "10" specifies one second of silence.

Text encoding:

```
vvarSpec = / hushSpec

; Interval of silence
hushSpec = "t=sil" ",v=" %d1-600
  ; Duration of silence, 100ms increments, 1 minute max.
```

6.3.6.11 Variable type: Tone

Definition: Plays an algorithmically generated tone. It is the counterpart of the INAP parameter "Tone".

Tone

–	toneId	Indicates the tone to be sent.
–	duration	Indicates the time duration in units of 100 milliseconds of the tone to be sent. ZERO indicates infinite duration.

Subtypes: Not applicable.

Value: Integer >= 0 (for both components of "Tone")

Text encoding:

```
vvarSpec =/ toneSpec
toneSpec = "t=tone" ",tid=" UINT32
           [",dur=" UINT32]
              ; specified in units of 100 milliseconds
```

6.4 Set extension to basic syntax

Package Name:	Announcement Set Syntax
PackageID:	setsyx (0x0049)
Description:	This package exists only to indicate that the MG is capable of processing the syntax described herein. An MGC learns that the capability is supported by auditing the packages supported by the termination on which playout is to be performed and verifying that this package is listed.

The syntax defined in this clause is used to select individual members of sets of media segments which convey equivalent semantic content.

Version:	2
Extends:	bannsyx version 1.

6.4.1 Properties

None.

6.4.2 Events

None.

6.4.3 Signals

None.

6.4.4 Statistics

None.

6.4.5 Procedures

Segment sets are described in 6.1.5. They provide an optional extension to the basic syntax for specification of a media segment. This Recommendation defines two selector tags:

6.4.5.1 "lang", the language selector

The values associated with this selector are the tags defined in IETF RFC 3066. These tags combine language with optional additional information such as region or country. Examples of such tags are "en-us" for English as spoken in the United States, or "cy" for Welsh (no locality qualifier required). The selector concept is applicable to multimedia content, although the examples provided in this Recommendation show its use only with audio segments.

6.4.5.2 "tatb", the selector for text attributes

The values associated with this selector are 16 bit unsigned integers. The used values have to be agreed between the Service Control part (INAP) and the equipment, where the Text To Speech applications are located. Variables of type spoken text or display text (see 6.5.5.1) can be qualified by this selector.

6.4.5.3 Text encoding for both selectors

Within an audio segment specification, all selectors must be specified in a single list of the form:

```
selList = "sel=" selSpec *( "&" selSpec )
```

Each selSpec names a selector type and assigns it one of its possible values.

```
selSpec = seltype "=" selval

seltype = "lang" / “tatb” / otherSel

otherSel = NAME
  ; As defined in Annex B/H.248.1.
  ; Selector types are case-insensitive.

selval = Language-Tag / UINT16 / otherSelVal
  ; Restriction: if seltype = “lang” then selval = Language-Tag,
  ; if seltype = “tatb” then selval = UINT16
```

The definition of Language-Tag is taken from IETF RFC 3066:

```
Language-Tag = Primary-subtag *( "-" Subtag )
   ; Case-insensitive

Primary-subtag = 1*8ALPHA
   ; Generally from ISO 639, but see RFC 3066

Subtag = 1*8(ALPHA / DIGIT)
   ; Generally from ISO 3166, but see RFC 3066
```

In general, selector values may be any combination of characters satisfying the safeChar production in Annex B/H.248.1, subject to the escaping rules applicable in the context of the segment specification.

```
otherSelVal = safeChar
```

6.4.5.3.1 Text encoding for provisioned segments

The specification of selector values, like the specification of embedded variable values, is done within the query part of an http: URL. To simplify parsing, selectors must be specified subsequent to any required embedded variable values. If any embedded variable values are present, the last variable value is separated from the first selector value by an ampersand. Thus the query production as it appears in 6.2.5.2.1 is extended as follows:

```
query =/ ( ( "var=" varVal *( "&var=" varVal) )  "&" selList
            ; embedded variable value(s) followed by selector
              specification(s)
       / selList
            ; selector specification(s) only
```

6.4.5.3.2 Text encoding for stand-alone variables

The definition of varSegSpec given in 6.3.6 is extended as follows:

```
varSegSpec =/ varSpec "&" selList
```

6.5 General text variable type extension to basic syntax

Package Name:	Phrase Variable Syntax
PackageID:	phrsyx (0x004a)
Description:	This package exists only to indicate that the MG is capable of processing the syntax described herein. An MGC learns that the capability is supported by auditing the packages supported by the termination on which playout is to be performed and verifying that this package is listed.

The syntax defined in this clause is used to designate phrase voice variables, which provide an arbitrary text to voice capability.

Version:	2
Extends:	vvsyx version 1.

6.5.1 Properties

None.

6.5.2 Events

None.

6.5.3 Signals

None.

6.5.4 Statistics

None.

6.5.5 Procedures

This clause defines the generic text variable type as an optional extension to the basic set of variable types defined in 6.3.6. See also 6.1.6.

6.5.5.1 Variable type: Phrase

Definition: Speaks or displays a specified phrase spelled out as a sequence of ASCII or UTF-8 characters according to the orthography of the language concerned. The variable may be qualified with the selector "text attributes".

Subtypes:

– **spk** (spoken text): default value.

– **dsp** (display text): With this subtype "Phrase" has the same syntax attributes, but stands for a text to be displayed on the end user's terminal.

Value:

Valid characters in the ASCII character set are a-z, A-Z, and blank (" "). Note that it is necessary to escape blanks as "%20" when present in the value of an embedded variable. Valid characters in any other character set may require further study.

Text encoding:

This clause extends the definition of vvarSpec given in 6.3.6 to include the generic text phrase variable type.

```
vvarSpec =/ phraseSpec
```

```
; Phrase
phraseSpec = "t=phrase" ",v=" [",s=spk"] [",s=dsp"]
    ( 1*(  ALPHA / ( %x20 / "%20" ) )
            ; ASCII string, restricted to (a-z, A-Z, blank)
            ; Note - need to escape blanks within embedded
                                      variables
    / ( "U+" 2*12HEX) *( "." 2*12HEX ) ) )
            ; General UTF-8 string as a sequence of dot-separated
            ; hex-encoded values introduced by "U+".
            ; HEX as defined in RFC 2396
; Case-sensitive since that may affect readout in some languages.
```

6.6 Examples

These examples use the Play signal of the Advanced Audio Server Base package, the PlayCollect signal of the AAS Digit Collection package, and the PlayRecord signal of the AAS Recording package.

Play an announcement that consists of a single segment residing on the Audio Server in a flat file:

```
Signals { aasb/play { an = "sid=<file://1947>" } }
```

Play an announcement that consists of a single segment residing on the Audio Server in a flat file using the http://localhost convention. This is exactly equivalent to the first example:

```
Signals { aasb/play { an = "sid=<http://localhost/1947>" } }
```

Play an announcement that consists of a single segment residing on the Audio Server in a hierarchical file system:

```
Signals { aasb/play { an = "sid=<file://audio/current/1947>" } }
```

Play an announcement that consists of a single segment residing on a machine named "darkstar" which is external to the Audio Server:

```
Signals { aasb/play { an = "sid=<http://darkstar/welcome>" } }
```

Play an announcement that consists of multiple segments. Line breaks added for clarity of presentation.

```
Signals { aasb/play { an = "sid=<file://audio/voice/brenda/123>,
                           sid=<file://audio/voice/althea/098>,
                           sid=<file://audio/voice/delia/086>" } }
```

Play an announcement that consists of a recording followed by a direct voice variable:

```
Signals { aasb/play {
              an = "sid=<file://gdtrfb>,var=<t=dat,s=mdy,v=19550809>" } }
```

Play an announcement which expresses the telephone number 0800 321 589 as "zero eight hundred ... three two one ... five eight nine":

```
Signals { aasb/play { an= "var=<t=dig,v=0>,var=<t=int,s=car,v=800>,
                          var=<t=sil,v=5>,var=<t=dig,v=321>,
                          var=<t=sil,v=5>,var=<t=dig,v=589>" } }
```

Play an announcement with two embedded variables. The variable values are given in the order in which they occur in the announcement:

```
Signals { aasb/play {
            an = "sid=<http://localhost/113?var=3999&var=20001015>"  } }
```

Play an announcement in English with a Glaswegian accent, assuming that http://localhost/1947 designates a set and set syntax is supported:

```
Signals { aasb/play { an ="sid=<  http://localhost/1947?sel=lang=en-gb-
glg>" } }
```

Play an announcement in Danish using a female voice. It is assumed that the announcement was provisioned in association with a selector of type "gender" with "female" as one of the possible values, as well as the "lang" selector type.

```
Signals { aasb/play {
      an ="sid=<http://localhost/jackstraw/ann45?sel=lang=da&
      gender=female>" }}
```

Play the first part of an announcement in English, the second part in the default language, and the third part in French. The first two segments are on the Audio Server, and the third segment is on a remote machine. Line breaks are added for clarity of presentation.

```
Signals { aasb/play { an ="sid=<http://localhost/ann1?sel=lang=eng>,
                         sid=<http://localhost/audio/myannouncements/ann2>,
                         sid=<http://darkstar/audio/ann3?sel=lang=fra>" } }
```

Play an announcement with a stand-alone date variable in English:

```
Signals { aasb/play { an = "sid=<http://darkstar/audio/ann7?sel=lang=en>,
                      var=<t=date,s=mdy,v=20001015&sel=lang=en>" } }
```

Play a prompt and collect an eight-digit password. If need be, play a reprompt, a no digits prompt, and a success or failure announcement. Give the user three attempts to enter the password. By default, password entry can interrupt prompting.

```
Signals { aasdc/playcol { ip = "sid=<file://enterpassword>",
                          rp = "sid=<file://tryagain>",
                          nd = "sid=<file://nodigits>",
                          sa = "sid=<file://goodpassword>",
                          fa ="sid=<file://badpassword>",
                          mxatt = 3 ,
                          dm = passwdmap  }
```

Play a prompt and record voice. If the user does not speak, play a no speech prompt. By default there is no success or failure announcement, and pre- and post- speech timers are each 5 seconds. Give the user two attempts to record. The recording may not be longer than 5 minutes and the recording is to have a segment identifier chosen by the MG.

```
Signals { aasrec/playrec { ip = "sid=<file://sayname>",
                           ns ="sid=<file://nospeech>",
                           mxatt = 2,
                           rlt = 3000,
                           rid = "$"
} }
```

Play an announcement ten percent faster than normal speed and five decibels softer than normal volume. Play the announcement three times with two seconds of silence between plays.

```
Signals { aasb/play { an = "sid=<file://brenda>",
                      sp = +10 , vl = -5 , it = 3 , iv = 20  } }
```

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.

```
Signals { aasdc/playcol { ip ="sid=<file://enterdigits>",
                          mxatt = 3, dm = elevendig, rsk = "*"   } },
DigitMap = elevendig { [0-1]xxxxxxxxxx }
```

7 New H.248.1 error codes

The packages of this Recommendation are based on a syntax, which has just been described, and rely on audio segment resources which are identified using that syntax. Errors in the execution of transaction requests may become apparent before the transaction response is returned, or may not appear until later. To enhance error reporting in the transaction response, this clause defines a number of application-specific error codes valid for the packages of H.248.9.

NOTE – The package documentation provides occasional guidance on the point at which errors should be detected for specific signals. However, this is to some extent implementation-dependent or even dependent on the specific resource: one implementation may assemble audio segments on a "just in time basis", discovering a missing segment in mid-play, while another ensures that all resources are present before beginning playout. A failure event is defined in the AAS Base package to provide for autonomous reporting of errors in the former case. In the latter, it seems reasonable to report any error in the transaction response.

7.1 Illegal syntax within an announcement specification

Error code #:	600
Name:	Illegal syntax within an announcement specification
Definition:	Some aspect of an announcement specification fails to conform to the required syntax.

Error text in the Error Descriptor: The offending portion of the specification.

Comment:	Indicates a possible software error at the MG or MGC.

7.2 Variable type not supported

Error code #:	601
Name:	Variable type not supported
Definition:	While the syntax of a stand-alone variable segment specification is apparently correct, the MG does not support the specified variable type.

Error text in the Error Descriptor: The offending segment specification.

Comment: The MGC can audit to determine the non-basic variable types supported by the MG.

7.3 Variable value out of range

Error code #: 602

Name: Variable value out of range

Definition: The value is syntactically correct but not acceptable. Applies to both embedded and stand-alone variables. Depending on implementation, this error may instead be reported by the Audio Operation Failure event.

Error text in the Error Descriptor: The offending segment specification.

Comment: Indicates possible provisioning error at the MG or MGC.

7.4 Category not supported

Error code #: 603

Name: Category not supported

Definition: The entity responsible for executing the query part of a provisioned audio segment has encountered a component category (e.g., "sel"), which it does not support. Depending on implementation, this error may instead be reported by the Audio Operation Failure event.

Error text in the Error Descriptor: The offending segment specification.

Comment: The MGC can audit to determine the non-basic categories supported by the MG.

7.5 Selector type not supported

Error code #: 604

Name: Selector type not supported

Definition: The tag following the "sel=" keyword is not provisioned as a selector type on the entity responsible for executing the query part of a provisioned audio segment. Depending on implementation, this error may instead be reported by the Audio Operation Failure event.

Error text in the Error Descriptor: The offending segment specification.

Comment: Indicates a probable provisioning error at the MG or MGC.

7.6 Selector value not supported

Error code #: 605

Name: Selector value not supported

Definition: The given value is not one, which is provisioned on the entity responsible for executing the query part of a provisioned audio segment. Depending on implementation, this error may instead be reported by the Audio Operation Failure event.

Error text in the Error Descriptor: The offending segment specification.

Comment: Indicates a probable provisioning error at the MG or MGC.

7.7 Unknown segment ID

Error code #: 606

Name: Unknown segment ID

Definition: A segment identified by a provisioned segment reference cannot be located. Depending on implementation, this error may instead be reported by the Audio Operation Failure event. See Error code 611 for the special case of failure of the MakePersistent operation.

Error text in the Error Descriptor: The offending segment specification.

Comment: Indicates a probable provisioning error at the MG or MGC.

7.8 Mismatch between play specification and provisioned data

Error code #: 607

Name: Mismatch between play specification and provisioned data

Definition: This error indicates a discrepancy between the contents of the query part of a provisioned segment specification and what has been provisioned for that segment. Depending on implementation, this error may instead be reported by the Audio Operation Failure event.

Error text in the Error Descriptor: The offending segment specification.

Comment: Indicates a probable provisioning error at the MG or MGC.

7.9 Provisioning error

Error code #: 608

Name: Provisioning error

Definition: For example, a provisioned segment identifier actually points to a sequence of physical segments, but one is missing. Depending on implementation, this error may instead be reported by the Audio Operation Failure event.

Error text in the Error Descriptor: The offending segment specification.

Comment: Indicates a probable provisioning error at the MG or MGC.

7.10 Invalid offset

Error code #: 609

Name: Invalid offset

Definition: The magnitude of the offset in a PlayCollect signal exceeds the actual length of the initial prompt. Since it is possible that the MG does not detect this condition before the transaction response is sent, this error may instead be reported by the Audio Operation Failure event.

Error text in the Error Descriptor: –

Comment: –

7.11 No free segment ids

Error code #: 610

Name: No free segment ids

Definition: The local space of segment identifiers is exhausted and the RecordingIdentifier parameter of the PlayRecord command was "$".

Error text in the Error Descriptor: –

Comment: –

7.12 Temporary segment not found

Error code #:	611
Name:	Temporary segment not found
Definition:	The MakePersistent signal failed because the target temporary segment was not associated with this termination.

Error text in the Error Descriptor: URI of the missing segment.

Comment: The segment may no longer exist because it timed out, or it may not have been recorded on this termination.

7.13 Segment in use

Error code #:	612
Name:	Segment in use
Definition:	A request to delete a persistent segment has failed because it is in use by another operation.

Error text in the Error Descriptor: The URI of the in-use segment.

Comment: –

8 Advanced audio server base package

Package Name:	Advanced audio server base package
PackageID:	aasb (0x0033)
Description:	The Advanced Audio Server (AAS) Base Package provides a signal to play an announcement and an event to indicate failure of the playout request. In connection with the latter, the package defines a return code and some possible values of that code. The aasb/play package cannot be applied to a termination unless it supports at least the Basic Announcement Syntax package.
Version:	1
Extends:	None.

8.1 Properties

None.

8.2 Events

8.2.1 Audio operation failure

Event name:	Audio operation failure
EventID:	audfail (0x0001)
Description:	Signifies the failure of an Advanced Audio Server operation subsequent to the return of the response to the transaction, which invoked it.

EventDescriptor parameters:

None.

ObservedEventDescriptor parameters:

Parameter Name:	Return Code
ParameterID:	rc (0x0001)
Description:	A return code indicating why an Advanced Audio Server operation failed.
Type:	Integer
Optional:	No
Possible values:	Failure return codes range from 600-699. Failure codes 600 to 612 report the same errors as the corresponding error codes in clause 7, except that the error conditions in the present case are detected after the transaction reply has been returned. (The possibility of such post-reply errors is dependent on the implementation and the specific audio segments invoked.)

600	Illegal syntax within an announcement specification
601	Variable type not supported
602	Variable value out of range
603	Category not supported
604	Selector type not supported
605	Selector value not supported
606	Unknown segment ID
607	Mismatch between play specification and provisioned data
608	Provisioning error
609	Invalid offset
610	No free segment ids
611	Temporary segment not found
612	Segment in use

In addition to these common error and failure codes, the following failure code values are defined in the base package. Additional code values may be added by other packages.

615	AAS hardware failure
616	AAS unspecified failure

Default: None

8.3 Signals

8.3.1 Play

Signal name:	Play
SignalID:	play (0x0001)
Description:	Plays one or more audio segments.
SignalType:	Defaults to BR (play continues until the specified or default number of iterations is completed).
Duration:	Not applicable to BR signals.

8.3.1.1 Additional parameters

8.3.1.1.1 Parameter name: Announcement

ParameterID:	an (0x0001)
Description:	An announcement to be played. Consists of one or more audio segments. This is the only non-optional parameter for the Play signal.
Type:	String
Optional:	No
Possible values:	A sequence of segment specifications adhering to the syntax described in clause 6. Support for optional elements of that syntax is indicated by the presence of the corresponding packages on the termination.
Default:	None

8.3.1.1.2 Parameter name: Iterations

ParameterID:	it (0x0002)
Description:	The maximum number of times an announcement is to be played.
Type:	Integer
Optional:	Yes
Possible values:	As described below, playout may end before the specified number of iterations is completed if the signal type is set to TO and the limit set by the Duration parameter is reached first. A value of 0 (zero) indicates that the announcement is to be repeated until halted by other means regardless of the number of iterations.
Default:	1

8.3.1.1.3 Parameter name: Interval

ParameterID:	iv (0x0003)
Description:	The interval of silence to be inserted between iterative plays. Specified in units of 10 milliseconds.
Type:	Integer
Optional:	Yes
Possible values:	0 upwards.
Default:	None

8.3.1.1.4 Parameter name: Speed

ParameterID:	sp (0x0004)
Description:	The relative playback speed of announcement specifiable as a positive (faster) or negative (slower) percentage variation from the normal playback speed. Actual playback speed as a percentage of normal speed is equal to the value of this parameter plus 100.
Type:	Integer
Optional:	Yes
Possible values:	–99 upwards.
Default:	0

8.3.1.1.5 Parameter name: Volume

ParameterID:	vl (0x0005)
Description:	The relative playback volume of announcement specifiable as a positive (louder) or negative (quieter) decibel variation from the normal playback volume.
Type:	Integer
Optional:	Yes
Possible values:	Implementation-dependent.
Default:	0

8.4 Statistics

None.

8.5 Procedure

The MGC invokes aasb/play with at least the announcement parameter set to play out a specified announcement. Announcement playout is subject to termination by events or new Signals descriptor settings in the normal way. If the signalType parameter is set to OO, this is the only way to end the announcement: the Duration and Iterations parameters are both ignored. If the signalType parameter is set to its default value of BR, Duration is ignored but the announcement will complete when the specified number of iterations has been played out. If the signalType parameter is set to TO, the announcement will complete at the earlier of the elapse of the amount of time given by the Duration parameter (which must be specified) and the completion of playout of the number of iterations and intervening pauses specified by the Iterations parameter.

The MGC can use the standard signal NotifyCompletion capability to determine when and why playout has ended. For more detailed information on failures, the MGC should enable the Playout Failure event.

The aasb/play signal can be used as part of a prompted digit collection operation. The MGC must either enable individual DTMF digit events or a standard H.248.1 digit map as well as invoking aasb/play. When individual DTMF digit events are enabled, the MGC can, if required, set the event KeepActive flag so that prompting continues to completion even if the subscriber starts keying early. If the MGC determines that the subscriber has made an error or has not keyed anything, the MGC can reinvoke the aasb/play signal with new prompts as required.

9 AAS Digit collection package

Package Name:	AAS Digit Collection Package
PackageID:	aasdc (0x0034)
Description:	The AAS Digit Collection Package extends the AAS Base Package by providing a signal and event to coordinate digit collection with the playout of prompting announcements. This provides an optimization over the use of aasb/play to collect digits, as described in clause 8. The use of aasdc/playcol avoids the messaging otherwise needed to invoke reprompts and to report digits not conforming to an expected pattern.
Version:	2
Extends:	aasb (0x0033) version 1

9.1 Properties

None.

9.2 Events

9.2.1 Audio operation failure

Event Name:	Audio Operation Failure
EventID:	audfail (0x0001)
Description:	This package adds the following codepoints for the return code returned by the Audio Operation Failure event defined in 8.2.1:

617 Premature termination of operation. The audio operation was terminated before its normal completion, by recognition of an event with the KeepActive flag not set, by replacement of the Signals descriptor without continuation of the signal, or by expiry of the signal duration timer.

618 Invalid command key sequence detected.

619 Max attempts exceeded. The final attempt collected digits, which did not match a pattern in the digit map.

620 No digits. The maximum number of attempts was reached and no digits were entered in the final attempt.

9.2.1.1 EventDescriptor parameters

None.

9.2.1.2 ObservedEventDescriptor parameters

See 8.2.1.

9.2.2 PlayCollect Success

Event Name:	Play Collect Success
EventID:	pcolsucc (0x0002)
Description:	This event signifies the successful completion of a playcol signal.

9.2.2.1 EventDescriptor parameters

None.

9.2.2.2 ObservedEventDescriptor parameters

9.2.2.2.1 Digits Collected

Parameter Name:	Digits collected
ParameterID:	dc (0x0003)
Description:	The DTMF digits that were collected during a play collect signal.
Type:	String
Optional:	No
Possible values:	Any sequence of valid DTMF digits 0-9, A-D or a-d, *, or #. A digit may be preceded by the long-duration modifier "Z" or "z" if detection of a long-duration tone in that position was enabled by the digit map named in the playcol signal.
Default:	None

9.2.2.2.2 Number of Attempts

Parameter Name:	Number of Attempts
ParameterID:	na (0x0002)
Description:	The number of attempts the MG made to collect a valid digit pattern.
Type:	Integer
Optional:	No
Possible values:	1 upwards.
Default:	None

9.2.2.2.3 Amount played

Parameter Name:	Amount played
ParameterID:	ap (0x0003)
Description:	The length played of the initial prompt, if that prompt was interrupted (i.e., by digit input when NonInterruptiblePlay was FALSE), in 10 ms units.
Type:	Integer
Optional:	Yes
Possible values:	0 upwards.
Default:	None

9.3 Signals

9.3.1 PlayCollect

Signal Name:	PlayCollect
SignalID:	playcol (0x0002)
Description:	Plays an announcement or tone (optionally) and collects dtmf digits or voice input by the user. The most complete model supported by playcol consists of an initial prompt, a reprompt if invalid digits are entered, a differing reprompt if the user fails to enter any digits at all, a success announcement played when a valid sequence of digits has been collected, and a failure announcement played if the attempt to collect digits fails. Defaults are assigned if particular announcements within this model are not specified, as indicated in the documentation of the individual parameters.
SignalType:	Defaults to TO
Duration:	Default as provisioned for the termination. Used only as a guard against excessive duration of the total collection operation.

9.3.1.1 Additional parameters

9.3.1.1.1 Initial Prompt

Parameter Name:	InitialPrompt
ParameterID:	ip (0x0001)
Description:	The initial announcement prompting the user to enter DTMF digits. May consist of one or more audio segments. If not specified, digit collection begins immediately.

Type:	String
Optional:	Yes
Possible values:	Any announcement specification conforming to the syntax described in clause 6. Support for optional aspects of that syntax, for this and the other announcement parameters, is indicated by the presence of the associated packages.
Default:	None

9.3.1.1.2 Reprompt

Parameter Name:	Reprompt
ParameterID:	rp (0x0002)
Description:	Played after the user has made an error such as entering an invalid digit pattern. Consists of one or more audio segments. Defaults to Initial Prompt.
Type:	String
Optional:	Yes
Possible values:	Any announcement specification conforming to the syntax described in clause 6.
Default:	None

9.3.1.1.3 Number of Digits Prompt

Parameter Name:	NoDigitsPrompt
ParameterID:	nd (0x0003)
Description:	Played after the user has failed to enter any digits following a prompt. Consists of one or more audio segments. Defaults to Reprompt.
Type:	String
Optional:	Yes
Possible values:	Any announcement specification conforming to the syntax described in clause 6.
Default:	None

9.3.1.1.4 Successful Announcement

Parameter Name:	SuccessAnnouncement
ParameterID:	sa (0x0004)
Description:	Played when data collection has succeeded. Consists of one or more audio segments. No default (i.e., no audio is played if this parameter is unspecified).
Type:	String
Optional:	Yes
Possible values:	Any announcement specification conforming to the syntax described in clause 6.
Default:	None

9.3.1.1.5 Announcement Failure

Parameter Name:	FailureAnnouncement
ParameterID:	fa (0x0005)
Description:	Played when all data entry attempts have failed. Consists of one or more audio segments. No default (i.e., no audio is played if this parameter is unspecified).
Type:	String
Optional:	Yes
Possible values:	Any announcement specification conforming to the syntax described in clause 6.
Default:	None

9.3.1.1.6 Non Interruptible Play

Parameter Name:	NonInterruptiblePlay
ParameterID:	ni (0x0006)
Description:	Specifies whether or not prompts are interruptible by digit input.
Type:	Boolean
Optional:	No
Possible values:	TRUE (prompts are non-interruptible) or FALSE (prompts are interrupted by digits).
Default:	FALSE

9.3.1.1.7 Keep Digits

Parameter Name:	KeepDigits
ParameterID:	kdg (0x0007)
Description:	Specifies handling of digits detected during the playout of a non-interruptible prompt. As described in 9.5.1, digits entered during a non-interruptible prompt will be accumulated if KeepDigits is TRUE.
Type:	Boolean
Optional:	Yes
Possible values:	TRUE or FALSE. Default is FALSE (digits detected during a non-interruptible prompt are ignored).
Default:	FALSE

9.3.1.1.8 Clear Digit Buffer

Parameter Name:	ClearDigitBuffer
ParameterID:	cb (0x0008)
Description:	If set to TRUE, the MG clears the digit collection buffer before playing any prompt.
Type:	Boolean
Optional:	No
Possible values:	TRUE or FALSE.

Default:	FALSE

9.3.1.1.9 Maximum number of attempts

Parameter Name:	MaxAttempts
ParameterID:	mxatt (0x0009)
Description:	The maximum number of attempts the user is given to enter a valid digit pattern.
Type:	Integer
Optional:	No
Possible values:	1 upwards.
Default:	1

9.3.1.1.10 Digit Map

Parameter Name:	DigitMap
ParameterID:	dm (0x000a)
Description:	The name of a digit map active on the termination.
Type:	String
Optional:	No
Possible values:	For text encoding, any string matching the NAME production. The equivalent for binary encoding would be the Name production of Annex A/H.248.1, but a Name is an arbitrary set of 16 bits and does not necessarily constitute a legal UTF-8 character. Hence the binary digit map name must be converted to a string of four hex characters before being passed in the DigitMap parameter.
Default:	None

9.3.1.1.11 Speed

Parameter Name:	Speed
ParameterID:	sp (0x000b)
Description:	The relative playback speed of each prompt specifiable as a positive (faster) or negative (slower) percentage variation from the normal playback speed. Actual playback speed as a percentage of normal speed is equal to the value of this parameter plus 100.
Type:	Integer
Optional:	Yes
Possible values:	–99 upwards.
Default:	0

9.3.1.1.12 Volume

Parameter Name:	Volume
ParameterID:	vl (0x000c)
Description:	The relative playout volume of each prompt specifiable as a positive (louder) or negative (quieter) decibel variation from the normal playback volume.
Type:	Integer

Optional:	Yes
Possible values:	Implementation dependent.
Default:	0

9.3.1.1.13 Offset

Parameter Name:	Offset
ParameterID:	off (0x000d)
Description:	Specifies the offset into the initial prompt at which to start playing. A positive offset is the offset going forward from the beginning of the prompt. A negative offset is the offset going backwards from the end of the prompt. Offsets are specified in 10 millisecond units. Offsets are useful when the MGC is controlling digit collection at an atomic level (i.e., using a very simple digit map and using aasdc/playcol to play prompts). An example of application is where the user hits a DTMF key, playcol matches the key and sends a PlayCollect Success event to the MGC which includes the digit value and the amount of the prompt already played, and the MGC decides to ignore the key and tells the Audio Server to resume playing at the point of interrupt. Another application is to allow the user to skip back and forward through a prompt.
Type:	Integer
Optional:	No
Possible values:	0, positive, or negative. The absolute value cannot exceed the length of the initial prompt.
Default:	0

9.3.1.1.14 Restart Key

Parameter Name:	RestartKey
ParameterID:	rsk (0x000e)
Description:	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 up to the point where the command sequence was entered, replay the prompt, and resume digit collection. The use of this key does not constitute an attempt to enter user input (i.e., it does not count against the number of attempts specified by the MaxAttempts parameter). Restart Keys are handled locally by the Audio Server and are not returned to the MGC.
Type:	String
Optional:	Yes
Possible values:	A sequence of one or more characters from the set 0-9, A-D or a-d, *, and # representing DTMF digits.
Default:	None

9.3.1.1.15 Re-input Key

Parameter Name:	Reinput Key
ParameterID:	rik (0x000f)

Description:	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 up to the point of input of the command sequence and resume digit collection. The use of this key does not constitute an attempt to enter user input (i.e., it does not count against the number of attempts specified by the MaxAttempts parameter). Reinput keys are handled locally by the Audio Server and are not returned to the MGC.
Type:	String
Optional:	Yes
Possible values:	A sequence of one or more characters from the set 0-9, A-D or a-d, *, and # representing DTMF digits.
Default:	None

9.3.1.1.16 Return Key

Parameter Name:	Return Key
ParameterID:	rtk (0x0010)
Description:	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 collection attempt and return the terminating key sequence to the MGC. During a recording, all digits except for the restart, reinput, and return keys (if defined) are ignored and become part of the recording.
Type:	String
Optional:	Yes
Possible values:	A sequence of one or more characters from the set 0-9, A-D or a-d, *, and # representing DTMF digits. Default is no sequence defined (may be overridden by provisioning).
Default:	None

9.3.1.1.17 Iterations

Parameter Name:	Iterations
ParameterID:	it (0x0011)
Description:	The maximum number of times an announcement is to be played. Identical to "Iterations" in signal "Play" of Advanced Audio Server Base Package. This parameter is also necessary in "PlayCollect" since the INAP does not see any difference between simple announcement and an announcement/prompting sequence.
Type:	Integer
Optional:	Yes
Possible values:	See "Play" in aasb.
Default:	1

9.3.1.1.18 Interval

Parameter Name:	Interval
ParameterID:	iv (0x0012)
Description:	The interval of silence to be inserted between iterative plays. Specified in units of 10 milliseconds. Identical to "Interval" in signal "Play" of Advanced Audio Server Base Package. This parameter is also necessary in "PlayCollect" since the INAP does not see any difference between simple announcement and an announcement/prompting sequence.
Type:	Integer
Optional:	Yes
Possible values:	See "Play" in aasb.
Default:	None

9.3.1.1.19 End Input Key

Parameter Name:	EndInputKey
ParameterID:	eik (0x0013)
Description:	This parameter indicates the digit used to signal the end of input. When the Maximum Number of Digits equals the Minimum Number of Digits, the End Input Key (could be present but) has no meaning. This parameter can be one or two digits. When the Maximum Number of Digits is greater than the Minimum Number of Digits the following applies: If the End Input Key is not present, the end of input is indicated: – when the inter-digit timer expires; or – when the number of valid digits received equals the Maximum Number of Digits. If the End Input Key is present, the end of input is indicated: – when the inter-digit timer expires; or – when the end input digit is received; or – when the number of valid digits received equals the Maximum Number of Digits. If the inter digit timer expires or the End Input Key is received **and** the number of valid digits received is less than the Minimum Number of Digits, the input is specified as being erroneous. This parameter corresponds to the INAP parameter 'endOfReplyDigit'.
Type:	Octet string (size (1..2))
Optional:	Yes
Possible values:	Implementation dependent
Default:	None

9.3.1.1.20 Include End Input Key

Parameter Name:	IncludeEndInputKey
ParameterID:	iek (0x0014)
Description:	By default the 'EndInputKey' is not included in the collected digits returned to the call agent. If this parameter is set to TRUE, then the 'EndInputKey' will be returned with the collected digits to the call agent.
Type:	Boolean
Optional:	Yes
Possible values:	TRUE or FALSE.
Default:	FALSE

9.3.1.1.21 Voice Information

Parameter Name:	VoiceInformation
ParameterID:	vi (0x0015)
Description:	Specifies how the ARF accepts voice input. The default value "dtmfonly" (DTMF only) means the ARF accepts DTMF digits only. The value "voiceonly" (voice only) means the ARF accepts speech input only. The value "dtmfandvoice" (DTMF and voice) means the ARF accepts DTMF and speech input and reports both: DTMF digits and spoken digits. The value "dtmfkills" (DTMF kills voice) means that the ARF accepts both kinds of input. When a DTMF digit is recognized, all digits collected by voice recognition are discarded and no further voice input is accepted.

This parameter corresponds to the INAP parameter "voiceInformation".

Type:	Enumeration	
Optional:	No	
Possible values:	– "dtmfonly"	(0),
	– "voiceonly"	(1),
	– "dtmfandvoice"	(2),
	– "dtmfkills"	(3)
Default:	"dtmfonly".	

9.3.1.1.22 Voice Back

Parameter Name:	VoiceBack
ParameterID:	vc (0x0016)
Description:	Specifies how recognized digits are played back to the user in case of voice recognition. The value "novoiceback " (no voice back) means that the digits are not played back. The value "stepbystep" (step by step) means that digits are played back as they are recognized. The value "atend" (at end) means that digits are played back to the user at the end of the input. The value "arfcontrolled" (controlled by ARF) means that the method of voice back is controlled by the ARF. The value "vbDTMF" (voice back DTMF) means that only DTMF digits are played back.

This parameter corresponds to the INAP parameter "voiceBack".

Type:	Enumeration
Optional:	Yes
Possible values:	– "novoiceback" (0), – "stepbystep" (1), – "atend" (2), – "arfcontrolled"(3), – "vbDTMF" (4)
Default:	"novoiceback"

9.3.1.1.23 INAP Prompt Timer

Parameter Name: InapPromptTimer

ParameterID: ipt (0x0017)

Description: The maximum amount of time to play and possibly replay an announcement or tone. Specified in units of 100 milliseconds. No default. As part of a PlayCollect signal, this parameter specifies the duration of the Initial Prompt. In case of a tone, the InapPromptTimer is mandatory. In case of an announcement, the InapPromptTimer is optional and can occur in addition to the parameters Iterations and Interval.

The following special handling has to be considered:

– When the *InapPromptTimer* equals zero and the *Iterations* not specified means the Initial Prompt has to be played indefinitely.

– When the *InapPromptTimer* and *Iterations* are specified in the same signal, it means the end of the Initial Prompt is either the end of the *InapPromptTimer* or of the end of the *Iterations*, whichever comes first.

The *InapPromptTimer* includes any specified intervals.

Type:	Integer
Optional:	Yes
Possible values:	0 or more tenths of seconds
Default:	None

9.4 Statistics

None.

9.5 Procedures

To use the PlayCollect signal effectively, the MGC must enable the PlayCollect Success event. It should also enable the Audio Operation Failure event if detailed information on the reason for failure is desired. (If not, and signal failure notification is required, the Generic Package signal completion event can be used.)

In typical use, the MGC will provide a digit map, which fully specifies one or more valid patterns for user input. This makes fullest use of the capabilities of the MG to handle command key sequence screening and automatic reprompting.

An alternative mode of usage is that suggested in the documentation of the offset parameter: the MGC specifies a digit map which is satisfied by any DTMF key, receives the digits one at a time, and restarts the PlayCollect signal with an offset equal to the amount already played out. If messaging between the MG and MGC is quick enough, the user hears the initial prompt as an almost-continuous audio playout. The only value in using PlayCollect rather than aasb/play in this case is the possibility of an automatic NoDigitsPrompt playout.

If the PlayCollect signal is invoked with signalType set to TO (the default), the MG must interpret the duration parameter as a limit on the entire duration of the digit collection operation, not on the length of time for playout of the initial prompt. If the timer expires before a valid digit sequence is collected, the signal completes with reason "Timed Out" and an Audio Operation Failure event is generated with return code 617. If signalType is set to BR or OO, the operation continues until interrupted by an event or change of Signals descriptor, or it completes, either with success or with failure. In the completion case, the signal completion method is "Normal Completion".

A command key sequence consists of a command (or escape) key optionally followed by zero or more keys. An application that defines more than one command key sequence will typically use the same command key (e.g.,*) for all command key sequences. Each key sequence must be unique with respect to any other key sequences. Applications may support additional command key sequences beyond <RestartKey>, <ReinputKey>, and <ReturnKey>.

To allow MG processing of command key sequences, applications must choose a command key that is not in any digit map. If a command key is encountered, digit map processing will stop and subsequent keys will be processed as a command key sequence until either a key sequence is recognized or until it is clear that a key sequence cannot be recognized, at which point error 618 "Invalid command key sequence detected" is returned.

The number of attempts parameter returned in the PlayCollect Success event may be used to enhance provisioning of the PlayCollect function.

The MGC must take care to set the KeepActive flag on any events it enables which are not intended to interrupt the PlayCollect operation. This applies particularly if the MGC enables either individual digit events or a digit map completion event. Such enabling is not required for the playCollect operation to complete successfully.

9.5.1 PlayCollect digit processing model

Digit collection is performed under the guidance of a digit map active on the termination and named by the corresponding parameter of the playcol signal. The model of digit processing is similar to that for ordinary digit maps, with two exceptions:

- the possibility of restarting the process through reprompts without MGC intervention;
- the possible detection and execution of command key sequences.

The Audio Server supports type-ahead by default. That is, digit detection and accumulation into the digit collection buffer for matching against command key sequences and against the digit map begins as soon as the playcol command becomes active. Type-ahead can be turned off by specifying that digit collection begins only after the initial prompt has been played out.

The detailed digit collection logic is as follows, where references to playcol parameters are enclosed in angle brackets <> to make them stand out. It relies on two logical buffers: a digit collection buffer which receives all digits keyed by the user, whether part of intended user input or a command sequence, and the current dial string accumulated against the digit map. The contents of the digit collection buffer can exceed one digit only if digit input is allowed during noninterruptible prompt playout or while accumulating a multi-digit command sequence, but the logic treats the general case.

1) The playcol command becomes active. Number of attempts is zero. Set "current prompt" to <InitialPrompt>. Clear the digit collection buffer (which will receive all digits, whether part of intended user input or a command sequence).

2) Collection loop. Increment number of attempts. Initialize digit map processing. If <ClearDigitBuffer> is TRUE, clear the digit collection buffer.

3) Process the appropriate one of the following three cases:

   a) <NonInterruptiblePlay> is TRUE:

      Begin current prompt playout. If <KeepDigits> is TRUE, retain current digit collection buffer contents and allow (further) digit accumulation during playout. If <KeepDigits> is FALSE, clear the digit collection buffer and ignore digits detected during playout. When playout of the current prompt is completed, go to step 4).

   b) <NonInterruptiblePlay> is FALSE and digit collection buffer is non-empty:

      Do not play the current prompt. Go immediately to digit processing (step 7).

   c) <NonInterruptiblePlay> is FALSE and digit collection buffer is empty:

      Begin playout of the current prompt. If a digit is detected during prompt playout, halt playout immediately and go on to digit processing (step 7). Otherwise fall through to next step.

4) Current prompt playout ends. Begin digit accumulation if not already started. Start initial digit timer for digit map.

5) If a digit is detected, go on to digit processing (step 7). Otherwise go to next step.

6) Check number of attempts. If it is equal to <MaxAttempts>, play <FailureAnnouncement> if one has been specified, exit and generate an Audio Operation Failure event with return code 620 "No Digits". Otherwise set current announcement to <NoDigitsPrompt> and return to step 2).

7) Digit processing. Process any digits accumulated in the digit collection buffer and succeeding digits as they arrive, matching them first against command key sequences and then against the digit map. For digit map processing the timer rules of 7.1.14/H.248.1 apply. If a <RestartKey> command sequence is recognized, go to step 8). If a <ReinputKey> command sequence is recognized, go to step 9). If a <ReturnKey> command sequence is recognized, go to step 10). If a failure to match the digit map is detected (no pattern fully matched), go to step 11). Finally, if digit map processing completes successfully (full match to a pattern), play <SuccessAnnouncement> if one has been specified, generate a PlayCollect Success event with the collected digits, and exit.

8) <RestartKey> command sequence is recognized. Decrement number of attempts, retain any digit accumulation buffer contents beyond the <RestartKey> command sequence, set "current prompt" to <InitialPrompt>, and return to step 2).

9) <ReinputKey> command sequence is recognized. Reinitialize digit map processing, retain any digit accumulation buffer contents beyond the <ReinputKey> command sequence, and return to step 7).

10) <ReturnKey> command sequence is recognized. Play <SuccessAnnouncement> if one has been specified, generate a PlayCollect Success event with the <ReturnKey> command sequence in place of any collected digits, and exit.

11) Failure to match digit map. Check number of attempts. If it is equal to <MaxAttempts>, play <FailureAnnouncement> if one has been specified, exit and generate an Audio Operation Failure event with return code 619 "Max Attempts Exceeded". Otherwise set current announcement to <Reprompt>, retain any digit accumulation buffer contents beyond the digits already processed (i.e., discarding the digit which "broke the pattern"), and return to step 2).

10 AAS recording package

Package Name:	AAS recording package
Package-ID:	aasrec (0x0035)
Description:	AAS Recording Package: extends the AAS Basic Playout Package by providing signals and events to coordinate the collection of recorded voice with the playout of prompting announcements.
Version:	1
Extends:	aasb (0x0033) version 1

10.1 Properties

10.1.1 Maximum temporary record life

Property Name:	Maximum temporary record life
PropertyID:	maxtrl (0x0003)
Description:	Determines the maximum life of a temporary recording, in seconds, following completion of recording. Recordings made by the PlayRecord signal are temporary unless explicitly made persistent using the MakePersistent signal. Temporary recordings are deleted at the earlier of expiry of maxtrl or destruction of the termination on which the recording was made.
Type:	Integer
Possible values:	1 upwards
Default:	None
Defined in:	TerminationState
Characteristics:	read/write

10.2 Events

10.2.1 Audio operation failure

Event Name: Audio operation failure

EventID: audfail (0x0001)

Description: This package adds the following codepoints for the return code returned by the Audio Operation Failure event defined in 8.2.1. Note that codes 617 and 618 are also supported by the AAS Digit Collection package.

617	Premature termination of operation. The audio operation was terminated before its normal completion, by recognition of an event with the KeepActive flag not set, by replacement of the Signals descriptor without continuation of the signal, or by expiry of the signal duration timer.
618	Invalid command key sequence detected.
622	No speech was collected after <MaxAttempts> prompts.
623	Out of storage.
624	Unable to delete temporary audio segment. Upon expiry of maxtrl, or destruction of the termination, a recorded audio segment which had not been made persistent could not be deleted. In the timeout case, the segment may be in use by another operation on the same termination.

10.2.1.1 EventsDescriptor Parameters

See 8.2.1.

10.2.1.2 ObservedEventsDescriptor Parameters

See 8.2.1.

10.2.2 PlayRecord success

Event Name:	PlayRecord success
EventID:	precsucc (0x0002)
Description:	Signifies the successful completion of a playrec signal.

10.2.2.1 EventsDescriptor Parameters

None.

10.2.2.2 ObservedEventDescriptor parameters

10.2.2.2.1 Amount Played

Parameter Name:	Amount played
ParameterID:	ap (0x0001)
Description:	The length played of the initial prompt, if that prompt was interrupted, in 10 ms units.
Type:	Integer
Optional:	No
Possible values:	0 upwards.
Default:	1

10.2.2.2.2 Number of attempts

Parameter Name:	Number of attempts
ParameterID:	na (0x0002)
Description:	The number of times the user was prompted to make a recording.
Type:	Integer
Optional:	No
Possible values:	1 upwards.
Default:	1

10.2.2.2.3 Recording Result

Parameter Name:	Recording result
ParameterID:	res (0x0003)
Description:	The particular way in which the recording process terminated successfully.
Type:	Enumeration
Optional:	No

Possible values:	"normal" (0): a temporary audio segment has been recorded, and end of speech was detected before the expiration of the RecordLengthTimer period. "trunc" (1): a temporary audio segment has been recorded, and it was truncated when the RecordLengthTimer period expired. "keyend" (2): the Return Key command key sequence was detected. No recorded audio has been retained.
Default:	None

10.2.2.2.4 Recording Identity

Parameter Name:	Recording id
ParameterID:	ri (0x0004)
Description:	A URI assigned to the physical segment recorded during a playrec signal. This parameter is returned only if the RecordingIdentifier parameter to the playrec signal has been set to the ANY wildcard, "$". If this is the case the Audio Server allocates a unique URI, associates it with the newly recorded segment, and returns it to the MGC. If the PlayRecord operation is terminated by the Return Key command key sequence, the URI is deallocated and this parameter must not be present in the event notification.
Type:	String
Optional:	Yes
Possible values:	Any physical segment identifier satisfying the syntax of 6.2.5.2. If the identifier is an http:// URI, it must not have a query part.
Default:	None

10.2.2.2.5 Recording Duration

Parameter Name:	Record duration
ParameterID:	rdur (0x0005)
Description:	The total length of the recorded audio segment in 10 ms units.
Type:	Integer
Optional:	Yes
Possible values:	0 upwards. If the operation was terminated by the Return Key sequence, rdur must not be present in the event notification.
Default:	None

10.3 Signals

10.3.1 PlayRecord

Signal Name:	PlayRecord
SignalID:	playrec (0x0002)
Description:	Plays a prompting announcement (optionally) and records voice input by the user. The most complete model supported by playrec is similar to that supported by aasdc/playcol, except that there is no recognition of invalid input. There is an initial prompt, a reprompt if the user fails to speak, a success announcement played when a recording has been successfully collected, and a failure announcement played if the attempt to collect a recording fails. Defaults

	are assigned if particular announcements within this model are not specified, as indicated in the documentation of the individual parameters. The RecordLengthTimer and RecordingIdentifier parameters must be specified. All other parameters are optional.
Signal Type:	Defaults to TO
Duration:	Defaults to 30000 (5 minutes) or as provisioned for the termination.

10.3.1.1 Additional parameters

10.3.1.1.1 Initial Prompt

Parameter Name:	InitialPrompt
ParameterID:	ip (0x0001)
Description:	The initial announcement prompting the user to speak for the record. May consist of one or more audio segments.
Type:	String
Optional:	Yes
Possible values:	Any announcement specification conforming to the syntax described in clause 6. Support for optional aspects of that syntax, for this and the other announcement parameters, is indicated by the presence of the associated packages on the termination. If not specified, the MG proceeds to the recording phase immediately.
Default:	None

10.3.1.1.2 No Speech Prompt

Parameter Name:	NoSpeechPrompt
ParameterID:	ns (0x0002)
Description:	Played after the user has failed to speak following a prompt. Consists of one or more audio segments.
Type:	String
Optional:	Yes
Possible values:	Any announcement specification conforming to the syntax described in clause 6. Defaults to InitialPrompt.
Default:	None

10.3.1.1.3 Success of Announement

Parameter Name:	SuccessAnnouncement
ParameterID:	sa (0x0003)
Description:	Played when recording has succeeded. Consists of one or more audio segments.
Type:	String
Optional:	Yes
Possible values:	Any announcement specification conforming to the syntax described in clause 6. No announcement is played if this parameter is unspecified.
Default:	None

10.3.1.1.4 Failure Announcement

Parameter Name:	FailureAnnouncement
ParameterID:	fa (0x0004)
Description:	Played when all recording attempts have failed. Consists of one or more audio segments.
Type:	String
Optional:	Yes
Possible values:	Any announcement specification conforming to the syntax described in clause 6. No announcement is played if this parameter is unspecified.
Default:	None

10.3.1.1.5 Maximum Number of Attempts

Parameter Name:	MaxAttempts
ParameterID:	mxatt (0x0005)
Description:	The maximum number of prompts the user is given to speak. Prompts resulting from use of <RestartKey> are not included. if <MaxAttempts> is reached, <FailureAnnouncement> is played out if specified and an Audio Operation Failure event is generated with return code 622 "No Speech".
Type:	Integer
Optional:	Yes
Possible values:	1 upwards
Default:	None

10.3.1.1.6 Pre-Speech Timer

Parameter Name:	PreSpeechTimer
ParameterID:	prt (0x0006)
Description:	The amount of time to wait for the user to initially speak. Specified in units of 10 milliseconds.
Type:	Integer
Optional:	No
Possible values:	1 upwards
Default:	None

10.3.1.1.7 Post Speech Timer

Parameter Name:	PostSpeechTimer
ParameterID:	pst (0x0007)
Description:	The amount of silence necessary after the end of the last speech segment for the recording to be considered complete. Specified in units of 10 milliseconds. Once the PostSpeechTimer period has elapsed, the MG plays out <SuccessAnnouncement> if it has been specified and generates a PlayRecord Success event indicating normal termination.
Type:	Integer

Optional:	No
Possible values:	1 upwards
Default:	None

10.3.1.1.8 Record Length Timer

Parameter Name:	RecordLengthTimer
ParameterID:	rlt (0x0008)
Description:	The maximum allowable length of the recording, not including pre or post speech silence. Specified in units of 10 milliseconds. Once the recording length exceeds (RecordLengthTimer – PostSpeechTimer), the MG plays out <SuccessAnnouncement> if it has been specified and generates a PlayRecord Success event indicating truncation of the recording. A value of 0 (zero) means there is no limit to the recording length. The recording is open-ended, and it is up to the application to manage the storage used by the recording.
Type:	Integer
Optional:	No
Possible values:	0 upwards
Default:	None

10.3.1.1.9 Recording Identifier

Parameter Name:	RecordingIdentifier
ParameterID:	rid (0x0009)
Description:	Specifies a URI to be assigned to the physical segment which is to be recorded by the playrec event. If this parameter is set to the CHOOSE wildcard, "$", the Audio Server will allocate the URI, associate it with the newly recorded segment, and return it to the call agent with the OperationComplete event. This parameter is mandatory.
Type:	String
Optional:	No
Possible values:	Either "$" or a physical segment identifier satisfying the syntax of 6.2.5.2. If the identifier is an http:// URL, it must not have a query part.
Default:	None

10.3.1.1.10 Speed

Parameter Name:	Speed
ParameterID:	sp (0x000a)
Description:	The relative playback speed of each prompt specifiable as a positive (faster) or negative (slower) percentage variation from the normal playback speed. Actual playback speed as a percentage of normal speed is equal to the value of this parameter plus 100.
Type:	Integer
Optional:	Yes
Possible values:	–99 upwards
Default:	0

10.3.1.1.11 Volume

Parameter Name:	Volume
ParameterID:	vl (0x000b)
Description:	The relative playback volume of each prompt specifiable as a positive (louder) or negative (quieter) decibel variation from the normal playback volume.
Type:	Integer
Optional:	Yes
Possible values:	Implementation dependent
Default:	0

10.3.1.1.12 Offset

Parameter Name:	Offset
ParameterID:	off (0x000c)
Description:	Specifies the offset into the initial prompt at which to start playing. A positive offset is the offset going forward from the beginning of the prompt. A negative offset is the offset going backwards from the end of the prompt. Offsets are specified in 10 millisecond units. Offsets are useful to allow the user to skip back and forward through a prompt, particularly when that prompt is actually a user recording being played back.
Type:	Integer
Optional:	Yes
Possible values:	0, positive, or negative. The absolute value cannot exceed the length of the initial prompt.
Default:	0

10.3.1.1.13 Restart Key

Parameter Name:	RestartKey
ParameterID:	rsk (0x000d)
Description:	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 recording made up to the point where the command sequence was entered, replay the prompt, and reattempt to detect and record speech. The reprompt forced by this key does not count against the number of attempts specified by the MaxAttempts parameter.
Type:	String
Optional:	Yes
Possible values:	A sequence of one or more characters from the set 0-9, A-D or a-d, *, and # representing DTMF digits. Default is no sequence defined (may be overridden by provisioning).
Default:	None

10.3.1.1.14 Reinput Key

Parameter Name:	ReinputKey
ParameterID:	rik (0x000e)
Description:	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 recording collected up to the point of input of the command sequence and reattempt to detect and record speech without playing a new prompt.
Type:	String
Optional:	Yes
Possible values:	A sequence of one or more characters from the set 0-9, A-D or a-d, *, and # representing DTMF digits.
Default:	None

10.3.1.1.15 Return Key

Parameter Name:	ReturnKey
ParameterID:	rtk (0x000f)
Description:	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 recording attempt and delete any speech recorded to this point, play <SuccessAnnouncement> if specified, and generate a PlayRecord Success event indicating that the operation was terminated by <ReturnKey>. During a recording, all digits except for the restart, reinput, and return keys (if defined) are ignored and become part of the recording.
Type:	String
Optional:	Yes
Possible values:	A sequence of one or more characters from the set 0-9, A-D or a-d, *, and # representing DTMF digits.
Default:	None

10.3.2 Make persistent

Signal Name:	Make persistent
SignalID:	makepers (0x0003)
Description:	Makes the temporary audio segment identified by the given URI into a persistent audio segment. If this is not done, the temporary audio segment will be deleted when the termination on which it was created is destroyed or the lifetime set by the aasrc/maxtrl property expires.
Signal Type:	Defaults to BR
Duration:	Not applicable for default type.

10.3.2.1 Additional parameters

10.3.2.1.1 Recording Identifier

Parameter Name:	Recording Identifier
ParameterID:	rid (0x0001)

Description:	Identifies the audio segment which is to be made persistent.
Type:	String
Optional:	No
Possible values:	A physical segment identifier satisfying the syntax of 6.2.5.2. If the identifier is an http:// URL, it must not have a query part.
Default:	None

10.4 Statistics

None.

10.5 Procedures

The logic for recording is much simpler than that for digit collection. The number of attempts begins at zero. Each time a prompt is played, the number of attempts is incremented. Reprompting occurs when no user speech is detected within the time interval set by <PreSpeechTimer>. The end of speech is recognized when the user stops speaking for the amount of time given by <PostSpeechTimer>.

If the MG recognizes a DTMF command key sequence, it takes the appropriate action:

- If <RestartKey> is detected, any recorded audio is deleted, the initial prompt is replayed without incrementing the attempt count, and the current attempt is restarted.
- If <ReinputKey> is detected, any recorded audio is deleted and the current attempt is restarted without reprompting.
- If <ReturnKey> is detected, any recorded audio is deleted. If the MG allocated the URI identifying the recorded audio segment the URI is deallocated. The <SuccessAnnouncement>, if any, is played and a PlayRecord Success event is generated indicating termination of the operation by <ReturnKey>.

Applications may support additional command key sequences beyond <RestartKey>, <ReinputKey>, and <ReturnKey>.

When an Audio Operation Failure is generated by the PlayRecord signal, any recorded speech is deleted and any URI allocated as a segment identifier by the MG is deallocated.

Persistent audio segments are global to the MG. Thus a persistent segment created at one termination can be referred to in a signal invoked on another. However, temporary audio segments may only be referred to in operations on the termination at which they were recorded.

Failure of the MakePersistent signal must be reported as an appropriate error code in the response to the transaction invoking it. That is, the response must not be returned to the MGC until the outcome of the MakePersistent operation is known.

11 Advanced audio server segment management package

Package Name:	Advanced audio server segment management package
PackageID:	aassm (0x0036)
Description:	The Advanced Audio Server Segment Management Package provides a mechanism to override, restore, and delete persistent audio segments. This package is defined on a special logical segment control termination rather than individual terminations over which announcements may be played. It relies on the requirement that the audio segment namespace be global to the MG.

The MGC overrides a provisioned physical segment by specifying an alternative persistent physical segment. The URI of the provisioned physical segment will then resolve to the overriding persistent physical segment. The overriding persistent audio can subsequently be deleted and the original provisioned audio can be restored.

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

Segment override could be used for a feature where a standard greeting is played to all customers calling a retail store. Occasionally the store manager may want to call a special number and record a temporary greeting that overrides the standard greeting, for instance a greeting that announces a sale or may be a seasonal greeting of some kind. When the greeting is no longer wanted, the manager can call the special number, cancel the temporary greeting, and restore the standard greeting.

This package does not rely on the Advanced Audio Server Base Package, hence does not extend it.

Version:	1
Extends:	None

11.1 Properties

11.1.1 AAS segment control termination name

Property Name:	AAS segment control termination name
PropertyId:	ctlnam (0x0001)
Description:	Name of the AAS Segment Control Termination, if any, supported by the MG.
Type:	ASN.1 type TerminationID or ABNF type terminationId, depending on the encoding in use.
Possible values:	As provisioned in the MG. The value MUST NOT contain a wildcard.
Default:	None
Defined in:	TerminationState on ROOT.
Characteristics:	Read only.

11.2 Events

None.

11.3 Signals

The Advanced Audio Server Segment Management Package provides three new signals.

11.3.1 Delete persistent

Signal Name:	Delete persistent
SignalID:	delpers (0x0001)
Description:	Deletes an identified persistent audio segment.

SignalType:	BR
Duration:	Not applicable.

11.3.1.1 Additional parameters

11.3.1.1.1 Segment Identifier

Parameter Name:	Segment Identifier
ParameterID:	sid (0x0001)
Description:	Identifies the audio segment which is to be deleted.
Type:	String
Optional:	No
Possible values:	A physical segment identifier satisfying the syntax of 6.2.5.2. If the identifier is an http:// URI, it must not have a query part.
Default:	None

11.3.2 Override audio

Signal Name:	Override audio
SignalID:	override (0x0002)
Description:	Overlays the specified provisioned audio segment with a different persistent audio segment. If an overlay for this segment is already in place, the new overlay replaces it.
SignalType:	BR
Duration:	Not applicable.

11.3.2.1 Additional parameters

11.3.2.1.1 Target Segment

Parameter Name:	Target Segment
ParameterID:	tgtsid (0x0001)
Description:	Identifies the segment which is to be temporarily replaced by a new segment.
Type:	String
Optional:	No
Possible values:	A physical segment identifier satisfying the syntax of 6.2.5.2. If the identifier is an http:// URI, it must not have a query part.
Default:	None

11.3.2.1.2 Overriding Segment

Parameter Name:	Overriding Segment
ParameterID:	oversid (0x0002)
Description:	Identifies the segment which is to be played out in place of the target segment.
Type:	String
Optional:	No

Possible values:	A physical segment identifier satisfying the syntax of 6.2.5.2. If the identifier is an http:// URI, it must not have a query part.
Default:	None

11.3.3 RestoreAudio

Signal Name:	RestoreAudio
SignalID:	restore (0x0003)
Description:	Removes a previously imposed overlay segment, so that subsequent references to the target segment play out the originally provisioned content.
SignalType:	BR
Duration:	Not applicable.

11.3.3.1 Additional parameters

11.3.3.1.1 Target Segment

Parameter Name:	Target Segment
ParameterID:	tgtsid (0x0001)
Description:	Identifies the segment from which any overlay is to be removed.
Type:	String
Optional:	No
Possible values:	A physical segment identifier satisfying the syntax of 6.2.5.2. If the identifier is an http:// URI, it must not have a query part.
Default:	None

11.4 Statistics

None.

11.5 Procedures

The transaction response for a request which includes signals of this package must not be returned until the outcome of the invoked operations is known. At that point, if an error occurs and one of the error codes defined in clause 7 is applicable, it should be used in the returned error descriptor.

Printed in Switzerland
Geneva, 2005