Internet Draft
Document: draft-ietf-speechsc-protocol-eval-02
Expires: October 2003
Version 02

B. Wyld
Editor
Eloquant

June 2003

SPEECHSC Protocol Evaluation

Status of this Memo

Abstract

This document is the Protocol Evaluation Document for the SPEECHSC Working Group. Section 3 provides the summary of the individual protocol comparisons (in the sections 4-N following) against the SPEECHSC requirements [1].

Table of Contents

1. Overview....2
2. Protocol Proposals....3
3. Protocol Evaluation Summary and Conclusion....3
4. Session Control : ôBeepö Compliance Evaluation (Jerry Carter) 3
4.1. General notes:....3
4.2. Analysis of General Requirements....4
4.3. Analysis of Duplexing and Parallel Operation Requirements....4
4.4. Analysis of additional considerations (non-normative).5
4.5. Analysis of Security considerations....5
4.6. Interaction Model....5
5. Session Control : ôSIPö Complience Evaluation (Rajiv Dharmadhikari)....5
5.1. Introduction....5

5.2. Analysis of General Requirements.....6
5.3. Analysis of Duplexing and Parallel Operation Requirements.....7
5.4. Analysis of additional considerations (non-normative).7
5.5. Analysis of Security considerations.....7
5.6. Other Criteria.....7
5.7. Interaction Model.....8
6. Session Control : ôRTSPö Complience Evaluation (Brian Wyld)8
6.1. General Introduction.....8

6.2. Analysis of General Requirements.....9
6.3. Analysis of Duplexing and Parallel Operation Requirements.....9
6.4. Analysis of additional considerations (non-normative).9
6.5. Analysis of Security considerations.....10
6.6. Interaction Model.....10
7. Session Control : ôWeb Servicesö Complience Evaluation (Stephane H. Maes).....10
7.1. General Notes:.....10
7.2. Analysis of General Requirements.....11
7.3. Analysis of Duplexing and Parallel Operation Requirements.....13
7.4. Analysis of additional considerations (non-normative)13
7.5. Analysis of Security considerations.....14
7.6. Interaction Model.....14
8. Resource Control : ôMRCPö Complience Evaluation (Sarvi Shanmugham).....14
8.1. General.....14
8.2. Analysis of TTS requirements.....15
8.3. Analysis of ASR requirements.....16
8.4. Analysis of Speaker Identification and Verification Requirements.....17
9. Resource Control : ôRTSPö Complience Evaluation (Brian Wyld) 18
9.1. General Introduction.....18
9.2. Analysis of TTS requirements.....18
9.3. Analysis of ASR requirements.....18
9.4. Analysis of Speaker Identification and Verification Requirements.....19
10. Security Considerations.....19
11. References.....19

1. Overview

This document provides the template for the content for the SPEECHSC Protocol Evaluation document.

This section will contain an overview of the process.

Section 2 contains a list of the proposed protocols submitted to WG. These protocols are in fact of 3 different natures:

- a generic service/resource access system (Web Services)
- æclassicÆ session control protocols for media channels (BEEP, SIP, RTSP), of which RTSP also provides resource control.
- a specific ASR/TTS resource control message set designed to be tunneled in a session control protocol (MRCP)

Section 3 provides a conclusion of the evaluations of the proposed

protocols against the Requirements and framework, and recommends a direction for the creation of the speechsc protocol.

Sections 4-7 provide the individual protocol evaluations for the protocols providing session control against the SPEECHSC requirements [1] that relate to these needs.

Sections 8 and 9 provide individual protocol evaluations for protocols providing resource control against the SPEECHSC resource control requirements.

2. Protocol Proposals

This section contains a list of the existing protocols submitted to the SPEECHSC WG for consideration by the deadline.

1. BEEP
2. SIP
3. RTSP
4. MRCP (initial submission)
5. Web Services

Each protocol section contains a review of the protocolÆs level of compliance to each of the SPEECHSC Requirements [1] as derived from the proposed protocol documents. The following key will be used to identify the level of compliancy of each of the individual protocols:

T = Total Complience. Meets the requirement fully.
P = Partial Compliance. Meets some aspect of the requirement.
P+ = Complience possible. Could meet the requirement with ônaturalö evolution of the protocol.
F = Failed Compliance. Does not meet the requirement.

3. Protocol Evaluation Summary and Conclusion

In summary, it appears that the decomposition of the problem into session control and resource control gives rise to:

- SIP is a good fit for session control for the speechsc requirements

- MRCP is a good start for resource control for the speechsc requirements

The conclusion for the protocol evaluation is therefore to:

- create a new speechsc specific resource control only protocol, based on MRCP
- use sessions that have been established by SIP (and defines a means of referring to these sessions)

4. Session Control : ôBeepö Compliance Evaluation (Jerry Carter)

4.1. General notes:

The BEEP protocol provides a general framework for establishing connections, defining new channels, negotiating security, and performing user authentication. Protocols build on beep must define a profile detailing how connections are established and must define

a set of messages which will be delivered using BEEP. The protocol is peer-to-peer although client-server style requests could be easily handled.

The following sub-sections compare each individual requirement against the protocol.

4.2. Analysis of General Requirements

4.2.1. Reuse existing protocols [5.1]

T: Beep is a published protocol, listed as RFC 3080 (http://www.ietf.org/rfc/rfc3080.txt).

4.2.2. Maintain Existing Protocol Integrity [5.2]

P: BEEP assumes that protocols, such as SpeechSC, will add messages.

Supporting multiple clients using TCP may require some effort.

4.2.3. Avoid Duplicating Existing Protocols [5.3]

T: Building SpeechSC over BEEP would allow the specification to focus on managing the ASR, media server, and SI/SV resources and the possible interactions between them. The operations for establishing connections and defining new channels would be handled by BEEP.

4.2.4. Protocol efficiency [5.4]

P+: BEEP imposes a small overhead (roughly 40 bytes per message). It provides a mechanism for supporting multiple communication channels over a single port. If grouping of requests is desired, this would need to be handled by grouping the SpeechSC messages.

4.2.5. Explicit invocation of services [5.5]

T: Though it is primarily a peer-to-peer protocol, BEEP may act as a traditional client server protocol.

4.2.6. Server Location and Load Balancing [5.6]

P+: This functionality is not provided by BEEP. This would need to be added as an extension.

4.2.7. Simultaneous services [5.7]

T: Multiple channels providing different services is possible. Each service is simply a message type which is passed to the server using BEEP.

4.2.8. Multiple media sessions [5.8]

F: BEEP assumes a 1:1 using TCP/IP.

4.3. Analysis of Duplexing and Parallel Operation Requirements

4.3.1. Duplexing and Parallel Operation Requirements [9]

P+: Parallel operations may be obtained using multiple channels. A message on one channel could potentially interrupt activity happening on the second. BEEP is very flexible allowing the server to implement whatever behavior is desired.

4.3.2. Full Duplex operation [9.1.1]

T: BEEP is a peer-to-peer protocol allowing full duplex communication on a single channel or parallel communication on multiple channels.

4.3.3. Multiple services in parallel [9.1.2]

P+: Multiple services may be run on separate channels. Merging or T-ing of RTP must be implemented by the server.

4.3.4. Combination of services

TBD

4.4. Analysis of additional considerations (non-normative)

TBD

4.5. Analysis of Security considerations

4.5.1. Security Considerations [11]

P+: BEEP offers a mechanism for managing security and user authentication.
SpeechSC requires managing multiple data streams and some form of unified authentication / security might be a goal. If so, BEEP security should be revisited with this in mind.

4.6. Interaction Model

TBC : TO BE COMPLETED : Analysis of the interaction model of the protocol during the ædataÆ phase (ie after session establishment) and its suitability for speechsc.

5. Session Control : ôSIPö Complience Evaluation (Rajiv Dharmadhikari)

5.1. Introduction

SIP is a protocol for initiating, modifying, and terminating multimedia sessions. The protocol is considered an IETF standard and its specifications can be found in [2]. The following sections provide a general statement with regards to the applicability of SIP as the control protocol for SPEECHSC.

5.1.1. SIP General Applicability

SIP is a pretty mature, well understood, and frequently used session establishment protocol. It has gone through multiple revisions in the IETF standard process. There are number of

commercial and public domain implementations of SIP that are available. Because of its close resemblance to HTTP and being a text based protocol, there are large number of SIP application developers available.

5.1.2. SIP Use in VOIP environment

SIP is already being used to establish and redirect RTP streams from various end points. The SPEECHSC requires a protocol for controlling ASR, TTS and SV resources. When these resources are deployed in a VOIP network that requires them to process media carried in RTP, the SIP protocol is used in lot of deployments. Rather than inventing a new control protocol and introducing operational aspects of the new protocol, SIP can be reused for controlling SPEECHSC resources.

5.2. Analysis of General Requirements

5.2.1. Reuse existing protocols [5.1]

T: SIP is an existing, widely used, and mature protocol defined in [2].

5.2.2. Maintain Existing Protocol Integrity [5.2]

T: Existing SIP methods and header fields will not be changed when SIP is used to control SPEECHSC resources. In case, if extensions are required, SIP allows carriage of custom payload in the body. This payload is understood only by UAs and it does not impact protocol integrity.

5.2.3. Avoid Duplicating Existing Protocols [5.3]

T: Lot of the requirements for SPEECHSC operation can easily be satisfied by SIP, e.g. establishing RTP streams or redirecting them. Without SIP, new SPEECHSC protocol will have to duplicate lot of session management functionality.

5.2.4. Protocol efficiency [5.4]

T: SIP is a very lightweight protocol when run over TCP or UDP. It leverages efficiency available in TCP and UDP protocols that have been around for over 20 years.

5.2.5. Explicit invocation of services [5.5]

T: SIP URI mechanism allows invocation of different services.

5.2.6. Server Location and Load Balancing [5.6]

P+: SIP employs standard DNS name resolution for locating resources. SIP itself does not provide load balancing features. Application level load balancers can be used to load balance SIP requests.

5.2.7. Simultaneous services [5.7]

T: SIP allows simultaneous invocation of different services. SIP

allows forking or splitting the same media stream to different end points as defined in [2].

5.2.8. Multiple media sessions [5.8]

T: SIP uses SDP to describe RTP stream characteristics. This allows the control of direction of RTP stream such as bi-directional or

uni-directional. SIP allows a UA to establish sessions with multiple UAs for the same session.

5.3. Analysis of Duplexing and Parallel Operation Requirements

5.3.1. Duplexing and Parallel Operation Requirements [9]

T: SPEECHSC resource is a SIP UA that can handle session requests from different UAs.

5.3.2. Full Duplex operation [9.1.1]

T: Each SIP UA consists of a UAC and a UAS. This allows for full duplex operation.

5.3.3. Multiple services in parallel [9.1.2]

T: SIP allows simultaneous invocation of different services. SIP allows forking or splitting the same media stream to different end points as defined in [2].

5.3.4. Combination of services

T: See 5.6.3. SIP UA can invoke different services and combine the results.

5.4. Analysis of additional considerations (non-normative)

TBD

5.5. Analysis of Security considerations

5.5.1. Security Considerations [11]

T: SIP protocol employs different authentication schemes that are widely used in IP based protocols.

5.6. Other Criteria

The following criteria were also defined by the evaluator of SIP.

5.6.1. Ability to establish session between SPEECHSC client and SPEECHSC resource

T: SIP User Agent can establish a session with another SIP User Agent.

5.6.2. Ability to terminate session by either SPEECHSC client or SPEECHSC resource

T: SIP User Agent can terminate a session with another SIP User Agent.

5.6.3. Support reliable sequencing and delivery between SPEECHSC client and SPEECHSC resource

P: SIP can be run over TCP or UDP. When run over TCP, this

requirement is easily satisfied. When run over UDP, SIP User Agent is required to implement logic to ensure reliable sequencing and delivery.

5.6.4. Ability for SPEECHSC client to coordinate SPEECHSC resources on different machines for a single session

T: SPEECHSC client can use SIP to establish SIP sessions with different machines.

5.6.5. Ability for SPEECHSC resource to handle multiple SPEECHSC clients

T: SPEECHSC resource is a SIP UA that can handle session requests from different UAs.

5.6.6. The SPEECHSC resource should be able to generate asynchronous events or unsolicited messages

T: SIP allows asynchronous events or unsolicited messages to be generated using SUBSCRIBE/NOTIFY mechanism.

5.6.7. The SPEECHSC client and resource should have ability for authenticating each other

T: SIP protocol employs different authentication schemes that are widely used in IP based protocols.

5.6.8. Ability to determine success or failure from both SPEECHSC client and SPEECHSC resource side

T: The protocol has following response codes: 200 for success, 3xx, 4xx, and 5xx for failure.

5.6.9. Support for versioning between SPEECHSC client and SPEECHSC resource

P+: This will require an additional header or element in the body of SIP message for versioning. The current version field is intended for SIP protocol version.

5.7. Interaction Model

Speechsc has certain needs related to the interaction model of the protocol during the ædataÆ phase (ie after session establishment). Specifically, speechsc will require that the resource server can send unsolicited messages/transactions to the resource client to return results and indicate events.

SIP messages in the data phase can flow in both directions (client to server as well as server to client). SIP INFO message can be used for this purpose. The SIP INFO is intended for mid-call message semantics. With this message, transactions can be initiated/defined by both ends.

SIP therefore has an interaction model suited to the speechsc model, which supports peer-peer messaging with a basic transactional symmetrical request/response model.

6. Session Control : ôRTSPö Complience Evaluation (Brian Wyld)

6.1. General Introduction

RTSP is an existing protocol, orientated towards audio playback and recording. As such, it has support for RTP session control, with SDP used for session description, and a message set allowing operation as a player/recorder with audio ôVCRö controls.

Only the session control is evaluated here (see later section for evaluation of the resource control elements)

The following sub-sections compare each individual requirement against the protocol.

6.2. Analysis of General Requirements

6.2.1. Reuse existing protocols [5.1]

T: RTSP/RTP/SDP would be reused.

6.2.2. Maintain Existing Protocol Integrity [5.2]

T: The extensions to RTSP to allow speechsc use would be in the spirit of the protocol, and would not break existing servers or clients.

6.2.3. Avoid Duplicating Existing Protocols [5.3]

T: Using RTSP would not recreate it.

6.2.4. Protocol efficiency [5.4]

T: RTSP is a text based protocol, but is relatively succinct as messages are specific to their operation.

6.2.5. Explicit invocation of services [5.5]

T: RTSP service invocation is sufficient.

6.2.6. Server Location and Load Balancing [5.6]

F: RTSP does not address this topic; however it can be used with other IETF protocols such as SLP or UDDI to do so.

6.2.7. Simultaneous services [5.7]

T: RTSP allows simultaneous invocation of services on the same or different control channel.

6.2.8. Multiple media sessions [5.8]

T: RTSP allows multiple media sessions.

6.3. Analysis of Duplexing and Parallel Operation Requirements

6.3.1. Duplexing and Parallel Operation Requirements [9]

T: RTSP allows session setup that should fulfill these requirements.

6.3.2. Full Duplex operation [9.1.1]

T: RTSP can create a full duplex session.

6.3.3. Multiple services in parallel [9.1.2]

T: RTSP can request multiple operations of the same type on the same session.

6.3.4. Combination of services

T: RTSP can request multiple operations of different types on the same session.

6.4. Analysis of additional considerations (non-normative)

TBD

6.5. Analysis of Security considerations

6.5.1. Security Considerations [11]

F: RTSP provides no specific security functionality at all, but depends on other IETF security protocols (as it uses TCP) to pre-validate and protect the sessions.

6.6. Interaction Model

Speechsc has certain needs related to the interaction model of the protocol during the ædataÆ phase (ie after session establishment). Specifically, speechsc will require that the resource server can send unsolicited messages/transactions to the resource client to return results and indicate events.

RTSP messages in the data phase can flow in both directions (client to server as well as server to client). Transactions can be initiated/defined by both ends. Currently most of the defined transactions are C-S; however there already exists an ANNOUNCE message transaction that is used to transit general content in both directions (and is in fact used by MRCP to transport its resource control messages).

RTSP has therefore an interaction model suited to the speechsc model, which supports peer-peer messaging with a basic transactional symmetrical request/response model.

7. Session Control : ôWeb Servicesö Complience Evaluation (Stephane H. Maes)

7.1. General Notes:

Speech engines (speech recognition, speaker, recognition, speech synthesis, recorders and playback, NL parsers, and any other speech processing engines (e.g. speech detection, barge-in detection etc) etc...) as well as audio sub-systems (audio input and output sub-systems) can be considered as web services that can be described and asynchronously programmed via WSDL (on top of SOAP), combined in a flow described via WSFL, discovered via UDDI and asynchronously controlled via SOAP that also enables asynchronous exchanges between the engines.

This solution presents the advantage to provide flexibility, scalability and extensibility while reusing an existing framework that fits the evolution of the web: web services and XML protocols [WS1]

According to the web services framework, speech engines (audio sub-systems, engines, speech processors) can be defined as web services
that are characterized by an interface that consists of some of the

following ports:
- "control in" port(s): It sets the engine context, i.e. all the settings required for a speech engine to run. It may include addresses where to get or send the streamed audio or results.
- "control out" port(s): It produces the non-audio engine output (i.e. results and events). It may also involve some session control exchanges.
- "audio in" port(s): It receives streamed input data.

- "audio out" port(s): It produces streamed output data.

Audio sub-systems can also be treated as web services that can produce streamed data or play incoming streamed data as specified by
the control parameters.

The "control in" or "control out" messages can be out-of-band or sent or received interleaved with "audio in or out" data. This can be determined in the context (setup) of the web services.

Speech engines and audio sub-systems are pre-programmed as web services and composed into more advanced services. Once programmed by the application / controller, audio-sub-systems and engines await
an incoming event (established audio session, etc...) to execute the
speech processing that they have been programmed to do and send the results as programmed.

Speech engines as web services are typically programmed to handle completely a particular speech processing task, including handling of possible errors. For example, as speech engine is programmed to perform recognition of the next incoming utterance with a particular
grammar, to send result to a NL parser and to contact a particular error recovery process if particular errors occur.

The following sub-sections compare each individual requirement against the protocol.

7.2. Analysis of General Requirements

7.2.1. Reuse existing protocols [5.1]

T: Web services are is a class of protocols (framework) widely studied and developed across numerous standard bodies like W3C, OASIS, WS-I, Liberty, Parlay and adapted to numerous deployment environments issues at IETF, OMA, 3GPP, 3GPP2, JCP, etcà As an entry point, we recommend consulting the work at W3C [WS1].

7.2.2. Maintain Existing Protocol Integrity [5.2]

T: Web services is an XML-based framework that is by definition extensible to support appropriate syntax and semantics.

Web services are bound on underlying transport protocols. Numerous such binding have been specified. Others are in development. By handling at SPEECHSC at the level of the
Web services framework, the integrity is maintained for:

- underlying transport protocols (to which the web service are bound (e.g. SOAP)
- web service framework

This does not prevent introducing bindings to new protocols if needed. For example, binding to SIP or BEEP could be advantageous for mobile deployments.

7.2.3. Avoid Duplicating Existing Protocols [5.3]

T: By definition, the web service framework can be specified to remote control any web service. Specified syntax can be limited to avoid duplicating remote control functionalities offered by other protocols.

At the same time, the extensibility inherent to the framework guarantees that it is possible to specify (standard) or define (application specific) remote control for other entities beyond the current scope of SPEECHSC.

In that context and in view of unifying the remote control framework exposed to an application developer or a system integrator, it may be of interest to provide remote control syntax for special entities like prompt player etcà

7.2.4. Protocol efficiency [5.4]

P+ to P: Web services are by definition more verbose protocols. Hence, at this stage this does not qualify work a T mark.

However work is in progress (e.g. OMA, JCP) to optimize the exchanges to handle:
- Client with limited resources
- Constrained bandwidth

These rely on protocol compression and optimization, caching and gateways.

As such the protocols qualify as P+.

In addition, based on the qualification of efficiency provided in [WS8], the web service framework proposed for SPEECHSC and described in [WS1] relies indeed on known efficient techniques:
- Asynchronous pre-programming of the engines as web services to reduce exchanges and avoid racing conditions
- Possibility to piggy back on response message if transported on optimized protocols like SIP or BEEP.
- state caching in the engines that are considered as stand-alone, pre-packaged and pre-programmed engines.
- etcà

7.2.5. Explicit invocation of services [5.5]

T: Web service is typically used in a client-server environment. Solutions exist for peer to peer (service to service) etcà

Web services have been deigned to support clients and servers at least one of which is operating directly on behalf of the user requesting the service.

In addition, work on-going at OMA and JCP addresses some of these issues in mobile environment with the introduction of possible web service gateways.

7.2.6. Server Location and Load Balancing [5.6]

T: Web services are widely developed for e-business applications. Numerous tools and mechanisms have been provided for service discovery ad advertisement. In addition, numerous offerings provide

routing and load balancing capabilities as part of the web application server used to deploy the web service.

Note that web services do not specify server location or load balancing; but they are deployed on systems that provide such functionalities. As web services are expected to be widely used in the future and central to most e-business offerings, it is to expect that such tools will become even more pervasive and efficient.

7.2.7. Simultaneous services [5.7]

Web services allow control (interface) and composition of web services at will (e.g. WSFL).

7.2.8. Multiple media sessions [5.8]

T: The framework proposed does not pre-supposes how many ports or streams are associated to the engine. Different inbound and outbound can be used at will

7.3. Analysis of Duplexing and Parallel Operation Requirements

7.3.1. Duplexing and Parallel Operation Requirements [9]

T: As explained, web services allow control (interface) and composition of web services at will (e.g. WSFL). Also, it does not pre-supposes how many ports or streams are associated to the engine. Different inbound and outbound can be used at will; in full duplex or even between engines as supported by WSFL [WS4] and WSXl [WS7].

7.3.2. Full Duplex operation [9.1.1]

T:

7.3.3. Multiple services in parallel [9.1.2]

T:

7.3.4. Combination of services

T: As explained, web services allow control (interface) and composition of web services at will (e.g. WSFL) into complex parallel, serial or coordinated combinations as supported by WSFL [WS4] and WSXl [WS7].

7.4. Analysis of additional considerations (non-normative)

The framework proposed supports:
- Use of SDP to describe sessions and streams for the streamed channels
- Time stamps could be transmitted as part of the control messages at the web service level or in band (e.g. with dynamic payload

switch or within the payload).
- The framework is compatible with any encoding scheme. This is illustrated by the work on SRF (Speech Recognition Framework) driven at 3GPP that supports conventional and DSR optimized codecs and possible exchange of speech meta-information (e.g. data that may be required to facilitate and enhance the server-side processing of the input speech and facilitate the dialog management in an automated voice service. These may include keypad events over-riding spoken input, notification that the UE is in hands-free

mode, client-side collected information (speech/no-speech, barge-in), etcà.).
- SOAP over SIP or BEEP to support the framework described in section 1 can also support VCR controls.
- real-time messaging between engine and control is supported within the framework (e.g. via SOAP or XML events). The framework also support exchange between engines (same process; see also WSXL [WS7]).

Although non-normative, the web service framework described probably deserves marks of P+ to T.

7.5. Analysis of Security considerations

7.5.1. Security Considerations [11]

Web services are evolving to provide security, authentication, encryption, trust management and privacy . Details can be found for example in [WS9] and explained in [WS10]. This is now an OASIS activity [WS11].

This framework would enable SPEECHSC to employ the security mechanism provided bu WS-Security for the remote control aspects. Exchanged media can rely on security mechanism at the transport / streaming level.

The web service framework described probably deserves marks of P+ to T.

7.6. Interaction Model

TBC : TO BE COMPLETED : Analysis of the interaction model of the protocol during the ædataÆ phase (ie after session establishment) and its suitability for speechsc.

8. Resource Control : ôMRCPö Complience Evaluation (Sarvi Shanmugham)

8.1. General

8.1.1. MRCP Framework and General Applicability

The overall MRCP framework, the components involved and their distribution and relationship to each other meet the framework specified by SPEECHSC. The primary advantage of MRCP is that it is a text based protocol designed to meet most of the requirements of SPEECHSC pertaining to speech recognition and Text to speech. Though Speaker Recognition (SR) and Speaker Verification (SV) are not supported in its current form, MRCP was explicitly designed to

be extendable for such needs. The core MRCP definition only deals with the control of the ASR or TTS resource and the commands and responses needed to achieve it.

There are multiple interoperable implementations of MRCP and hence is a proven technology. It leverages existing W3C XML standards for exchange of data between the client and the server resource. For Example, its uses the W3C XML grammar format (GRXML) along with W3C semantic attachments and Natural Language Semantic Markup Language

to exchange data with speech recognition resource. The W3C Speech Markup Language is used when dealing with Text to speech engines.

It was designed to work as a tunneled protocol, over RTSP or SIP. Hence it depends on the carrier protocol to establish a control and a media path between the client and the ASR or TTS server resource. Hence it gets most of the security and media pipe management operations for free. Once these are established, MRCP commands and responses are tunneled over, controlling the ASR or TTS resource on the server.

8.1.2. MRCP can be evolved

Though MRCP directly meets many of the needs of SPEECHSC. The notion that it is a tunneled protocol disallows its independent operation. Further more the tunneled aspect is also a less efficient protocol design.

But these can be addressed and the core MRCP messages can be evolved to either become standalone protocol by itself or extensions to an existing protocol such as SIP or RTSP. To make this a standalone protocol and allow MRCP to operate by itself, new session and media management messages need to be defined to allow it to operate independently. To evolve MRCP as extensions to SIP or RTSP would also be relatively simple since it is also a text based protocol with message format and headers very similar to them. In this protocol evaluation, the compliance evaluates MRCP from the perspective of evolution in one of these forms.

The following sub-sections compare each individual requirement relating to resource management against the protocol.

8.2. Analysis of TTS requirements

8.2.1. Requesting Text Playback [6.1]

T: MRCP has the SPEAK method for the client to request the TTS resource to playback text as an audio stream.

8.2.2. Text Formats [6.2]

T: When the client requests the TTS resource to playback a text stream it can provide the content in the following formats and through the following mechanism.

1. Plain text
2. W3C XML based Speech Markup Language (SSML)
3. This content to be spoken can be provided by value directly through the control path.
4. It also supports passing the content by reference. This is achieved having an audio tag inside the SSML markup text.

This URL is then fetched and played on the RTP stream in sequence with the rest of the text according to the SSML specification.

When the client sends plain text, SSML or another format of speech text the content is coded as a mime-type. Hence the server knows what format the speech content is coded in, and does not have to figure it out from the content.

8.2.3. Plain text [6.2.1]
T: see above

8.2.4. SSML [6.2.2]
T: see above

8.2.5. Text in Control Channel [6.2.3]
T : see above

8.2.6. Document Type Indication [6.2.4]
T: see above

8.2.7. Control Channel [6.3]
T: In MRCP, this Reset-Audio-Channel header defined for the ASR resource allows the recognizer to re-initialize the audio characteristics that it has learnt till then. This allows a recognizer resource to be used for multiple recognition sessions. It can be used for short single utterance recognitions as well. This is by applying the Reset-Audio-Channel header to every recognition. I suspect the performance may not be as good, due to the lack of line characteristics, but this is a recognizer issue.

8.2.8. Playback Controls [6.4]
T: MRCP supports the CONTROL method with the Jump-Target header can used to achieve, jumping in time or to an exact or relative location. It supports jumping in paragraphs, sentences, words and to specific markers that may be embedded in the speech content. The CONTROL method can be used with the Voice and Prosody parameters, derived from SSML, and can address the speed of speech or increasing/decreasing the volume. It also supports the PAUSE/RESUME methods to pause or resume a current SPEAK request.

8.2.9. Session Parameters [6.5]
T: As mentioned the previous section, MRCP supports voice and prosody parameters which are directly derived from the W3C SSML specification. These headers can be sent using the SET-PARAMS method and applied as a default for the entire session. They can also be applied in SPEAK requests to apply per usage or in the CONTROL message to change the parameters of an active SPEAK request.

8.2.10. Speech Markers [6.6]
T: Specifying speech markers in the content is supported through SSML. The CONTROL message can then be used to jump to specific marker points in the text. Also, when the TTS resource reaches specific markers in the text, the server would generate the SPEECH-MARKER method to the client.

8.3. Analysis of ASR requirements

8.3.1. Requesting Automatic Speech Recognition [7.1]

T: The client uses the RECOGNIZE method in MRCP to request the recognition resource to process the audio stream in the pipe. The RECOGNIZE method also specifies parameters and grammars the recognizer should match against.

8.3.2. XML [7.2]

T: Similar to the TTS resource in MRCP, ASR also uses XML data to exchange information between the client and the recognition resource. It supports the W3C GRXML to pass grammars from the client to the server. When the server is done recognizing, it uses the W3C Natural Language Semantic Markup Language (NLSML) to pass the results back to the client. It supports other grammar formats as well, as long as the server allows it. This is possible since, it uses mime-types to package this data and hence the format type is specified.

8.3.3. Grammar Specification [7.3.1]

P+: MRCP supports specifying the grammar both by value and by reference. The RECOGNIZE method can carry with it grammar content and/or a URI referring to the grammar content. Since MRCP supports referring a grammar, the referred grammar could be located on the server itself. With respect to sharing of grammars, the grammars defined/compiled through the DEFINE-GRAMMAR primitive are not sharable across sessions on the same server. This needs to be addressed to meet this set of requirements in full.

8.3.4. Explicit Indication of Grammar Format [7.3.2]

P+: see above

8.3.5. Grammar Sharing [7.3.3]

TBD

8.3.6. Session Parameters [7.4]

T: This requirement as defined is already fully met since MRCP is the referred standard for compliance.

8.3.7. Input Capture [7.5]

T: This is achieved by setting the Waveform-url header in the RECOGNIZE method. This tells the server to record the audio of the recognition and will return a URI to the client in the completion event, which can be used to retrieve or play back the audio.

8.4. Analysis of Speaker Identification and Verification Requirements

8.4.1. Requesting SI/SV [8.1]

F: not supported

8.4.2. Identifiers for SI/SV [8.2]

F: not supported

8.4.3. State for multiple utterances [8.3]

F: not supported

8.4.4. Input Capture [8.4]

F: not supported

8.4.5. SI/SV functional extensibility [8.5]

F: not supported

9. Resource Control : ôRTSPö Complience Evaluation (Brian Wyld)

9.1. General Introduction

RTSP is an existing protocol, orientated towards audio playback and recording. As such, it has support for RTP session control, with SDP used for session description, and a message set allowing operation as a player/recorder with audio ôVCRö controls. This comparison only addresses the existing resource control elements here and their applicability to the speechsc requirements.

The current PLAY state machine is exactly as required for TTS operation. Although by analogy RECORD could initiate an ASR session, with headers giving the grammer source or references, itÆs state machine is not nearly as compatible, and not at all for SV/SI.

9.2. Analysis of TTS requirements

9.2.1. Requesting Text Playback [6.1]

P+: the RTSP PLAY message semantics would require minor extensions

9.2.2. Text Formats [6.2]

P+: Text can be defined as all text types.

9.2.3. Plain text [6.2.1]

T: Plain text may be carried directly in the message payload.

9.2.4. SSML [6.2.2]

T: Text may be in any format.

9.2.5. Text in Control Channel [6.2.3]

T: Text may be attached to the control messages.

9.2.6. Document Type Indication [6.2.4]

T : Via the Content-Type header

9.2.7. Control Channel [6.3]

T: RTSP sessions may use a private or shared TCP connection.

9.2.8. Playback Controls [6.4]

T: RTSP defines playback control messages and a state machine.

9.2.9. Session Parameters [6.5]

T: RTSP defines operations for session parameter control.

9.2.10. Speech Markers [6.6]

P+: Markers may be inserted in the text, but to provide the required asynchronous events when a marker is synthesized will require use specific ANNOUNCE type messages for server->client

notification.

9.3. Analysis of ASR requirements

9.3.1. Requesting Automatic Speech Recognition [7.1]

P: The RECORD message and semantics could be used but would require extensions (stretching the current semantic quite a lot)

9.3.2. XML [7.2]
P+: Text can be defined as all text types.

9.3.3. Grammar Specification [7.3.1]
P+: Text can be defined as all text types.

9.3.4. Explicit Indication of Grammar Format [7.3.2]
T : Via the Content-Type headers

9.3.5. Grammar Sharing [7.3.3]
F: TBD

9.3.6. Session Parameters [7.4]
T: RTSP defines operations for session parameter control.

9.3.7. Input Capture [7.5]
P+: would require addition of a header to the initiation message.

9.4. Analysis of Speaker Identification and Verification Requirements

9.4.1. Requesting SI/SV [8.1]
F: not supported

9.4.2. Identifiers for SI/SV [8.2]
F: not supported

9.4.3. State for multiple utterances [8.3]
F: not supported

9.4.4. Input Capture [8.4]
F: not supported

9.4.5. SI/SV functional extensibility [8.5]
F: not supported

10. Security Considerations

Security considerations for the SPEECHSC protocol are covered by the comparison against the specific Security requirements in the SPEECHSC requirements document [1].

11. References

[1] Oran, D., "Requirements for Distributed Control of ASR, SI/SV and TTS Resources", draft-ietf-speechsc-reqts-04, June 6, 2003, work in progress.

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J.

Peterson, R. Sparks, M. Handley, E.Schooler, SIP: Session Initiation Protocol, RFC3265, June 2002. (Obsoletes RFC2543)

[WS1] W3C Web Services, http://www.w3c.org/2002/ws/
[WS2] Simple Object Access Protocol (SOAP) http://www.w3c.org/2002/ws/
[WS3] Web Services Description Language (WSDL 1.1), W3C Note 15 March

2001, http://www.w3.org/TR/wsdl.
[WS4] Leymann, F., Web Service Flow Language, WSFL 1.0, May 2001, http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
[WS5] UDDI, http://www.uddi.org/specification.html
[WS6] W3C Voice Activity, http://www.w3c.org/Voice/
[WS7] WSXL - Web Service eXperience Language submitted to OASIS WSIA and WSRP - WSXL - Web Service eXperience Language submitted to OASIS WSIA and WSRP
[WS8] Requirements for Distributed Control of ASR, SI/SV and TTS Resources, draft-ietf-speechsc-reqts-01.txt
[WS9] Security in a Web Services World: A Proposed Architecture and Roadmap, April 7, 2002, Version 1.0, http://www.verisign.com/wss/wss.pdf
[WS10] Kapil Apshankar, WS-Security, Security for Web Services, http://www.webservicesarchitect.com/content/articles/apshankar04.asp
[WS11] OASIS Web Services Security TC, http://www.oasis-open.org/committees/wss/

AuthorÆs Address

Brian Wyld
Eloquant SA
ZA Malvaisin
Le Versoud, France
Phone: +33 476 77 46 92
Email: brian.wyld@eloquant.com

Full Copyright Statement