A

MRCP Version 1

MRCP Version 1 (MRCPv1) is the predecessor to the MRCPv2 protocol. MRCPv1 was developed jointly by Cisco, Nuance and Speechworks, and is published under RFC 4463 [13]. MRCPv1 is an Informational RFC and therefore not a candidate for an IETF Internet Standard (the IETF published MRCPv1 for its historical value as an ancestor to the MRCPv2 standard). That said, MRCPv1 has been widely implemented by both speech server vendors and network equipment providers alike.

This appendix describes MRCPv1 and how it differs from MRCPv2. Many of the chapters in this book are relevant to MRCPv1. In Part I, Chapters 1 and 2 apply to both protocol versions. In Chapter 3, note that MRCPv1 only supports the speech synthesiser and speech recogniser resources, and that the Real Time Streaming Protocol (RTSP) replaces the Session Initiation Protocol (SIP) as the protocol between the MRCP client and media resource server. In Part II, Chapters 4 and 5 related to SIP are not relevant. Chapter 6 does apply to MRCPv1 as does Chapter 7, subject to the caveats described in Section A.3 below. All of the chapters in Part III are relevant to MRCPv1 (Sections 10.4 and 10.5 in Chapter 10 can be omitted, however). In Part IV of the book, only Chapters 12 and 13 are relevant to MRCPv1, subject to the differences described in Sections A.4 and A.5 below. Finally, in Part V, Chapter 16 on VoiceXML is relevant to MRCPv1 as is Chapter 17 on VoiceXML-MRCP interworking, with the exception that MRCPv1 does not support a resource for recording audio.

A.1 Overview

Loosely speaking and in terms of functionality, MRCPv1 can be thought of as a subset of MRCPv2. MRCPv1 only caters for two resource types: a speech synthesiser resource and a speech recogniser resource. This is in contrast to MRCPv2, which includes the speech recorder resource, and speaker verification and identification resource types. The features available in the two resources supported in MRCPv1 are also a subset of their MRCPv2 counterparts - for example, the speech recogniser resource in MRCPv1 does not support voice enrolled grammars or hotword.

Speech Processing for IP Networks **Dave Burke**

Another fundamental difference between MRCPv1 and MRCPv2 stems from the mechanisms used for media session management and protocol transport. In MRCPv1, the Real Time Streaming Protocol (RTSP) [54] is used to initiate and manage a media session with a media resource. MRCPv2, in contrast, employs SIP [2] for this purpose. With MRCPv1 the actual MRCP protocol messages are tunnelled through RTSP, that is, the MRCP message is carried within the message body of an RTSP message. This is in stark contrast to MRCPv2, which uses a dedicated connection-oriented control channel (such as TCP) for this purpose. Indeed, MRCPv1's use of RTSP is awkward and limited, and is certainly one motivation for the MRCPv2 protocol. MRCPv2 employs an architecturally cleaner and more flexible approach to session management and protocol transport.

A.2 Session Management and Message Transport

RTSP is intended as a control protocol for the delivery of real-time media from a streaming server. RTSP is commonly used today in desktop media players for establishing media streams with streaming servers that can deliver multimedia presentations over the Internet. RTSP's control machinery includes the ability to play and record media clips, pause and resume, jump to a different position, etc. MRCPv1 leverages RTSP's capability to manage media sessions but does not actually use any of RTSP's control machinery - the control mechanisms are instead contained within the MRCP messages. MRCPv1 reuses an RTSP message (`ANNOUNCE`) for tunnelling MRCP messages (note that the original purpose of `ANNOUNCE` was to post the description of a media presentation to the streaming server and not as a conduit for other protocols to piggy-back upon).

RTSP is a text-based protocol with a similar structure to that of SIP or HTTP. RTSP may run over TCP or UDP although it is most commonly used with TCP. MRCPv1 requires that media resource servers support TCP for transporting RTSP messages. In total, only four RTSP methods are used to support MRCPv1: `SETUP`, `TEARDOWN` and `DESCRIBE` for session management, and `ANNOUNCE` for tunnelling MRCP messages. We describe the RTSP session management mechanisms applied to MRCPv1 next.

An MRCPv1 media resource is identified by its RTSP URI, for example:

```
rtsp://example.com/media/speechsynthesizer/
rtsp://example.com/media/speechrecognizer/
```

To initiate a session to a media resource, an RTSP `SETUP` request is sent to the media resource server. MRCPv1 actually extends the semantics of RTSP around the usage of SDP by using the offer-answer model described in an earlier version of SIP (described in RFC 2543). The SDP offer is placed in the message body of the `SETUP` request and the the 200 OK response contains the selected media for the session (from the point of view of setting up the media session, this mechanism parallels SIPs `INVITE` - `200 OK` - `ACK` approach, where the SDP offer is placed in the `INVITE` message and the SDP answer is placed in the `200 OK` response). A session is destroyed via the RTSP `TEARDOWN` message. Figure A.1 illustrates how the RTSP messages `SETUP` and `TEARDOWN` are used to create and subsequently to destroy an MRCPv1 session. The corresponding messages follow.

F1 (client → server):

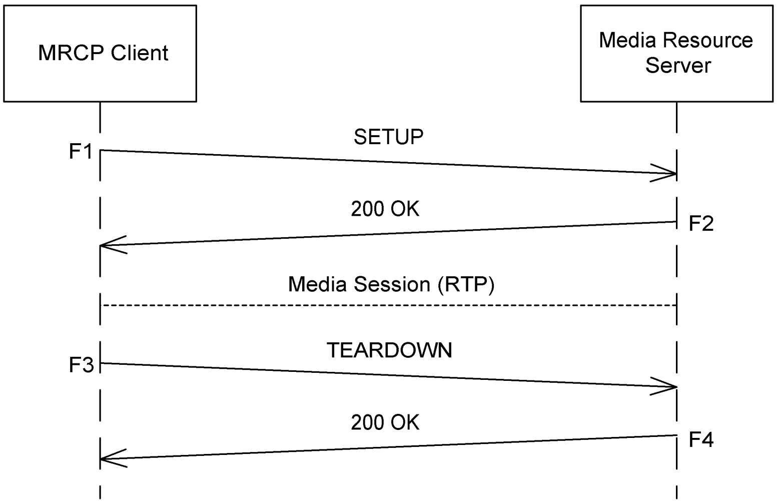

Figure A.1 RTSP flow for initiating and terminating a media session.

```
SETUP rtsp://example.com/media/speechsynthesizer/ RTSP/1.0
CSeq: 1
Transport: RTP/AVP;unicast;client_port=4588-4589
Content-Type: application/sdp
Content-Length: 182

v=0
o=client 2890844527 2890844527 IN IP4 10.0.0.1
s=-
c=IN IP4 10.0.0.1
t=0 0
m=audio 4588 RTP/AVP 0 96
a=rtpmap:0 pcmu/8000
a=rtpmap:96 telephone-event/8000
a=fmtp:96 0-15
```

F2 (server → client):

```
RTSP/1.0 200 OK
CSeq: 1
Session: 12345678
Transport: RTP/AVP;unicast;
           client_port=4588-4589;server_port=6256-6257
Content-Type: application/sdp
Content-Length: 129

v=0
```

```
o=server 2890844527 2890844527 IN IP4 10.0.0.2
s=-
c=IN IP4 10.0.0.2
t=0 0
m=audio 6256 RTP/AVP 0
a=rtpmap:0 pcmu/8000
```

F3 (client → server):

```
TEARDOWN rtsp://example.com/media/speechsynthesizer RTSP/1.0
CSeq: 4
Session: 12345678
```

F4 (server → client):

```
RTSP/1.0 200 OK
CSeq: 4
```

The RTSP syntax looks somewhat similar to that of SIP. Each request has a corresponding response with a status code (where values of 2xx indicate success). The CSeq header field plays a similar role to that of SIP's equivalently named header by indicating a sequence number for an RTSP request-response pair. The Transport header field indicates the transport protocol to be used (in this case RTP) and provides some additional information such as the client and server RTP/RTCP port pairs (this information is also available in the SDP). The media for the session is negotiated through the SDP offer in the SETUP message and SDP answer in the 200 OK response (again, this is an MRCPv1 extension since RTSP typically would not include SDP in the SETUP message or expect SDP in the response). The response to the SETUP request also includes a Session header field that must be specified in all subsequent messages from the client for this session. The client can invoke another resource on the same server to be used in the same session (e.g. a speech recogniser in this example) by issuing a second SETUP request and including the same Session value returned from the first one. This is analogous to issuing a SIP re-INVITE in MRCPv2. For example, by issuing two SETUP requests, one for a speech synthesiser resource and one for a speech recogniser resource, and using the same Session value, both resources can share the same bi-directional media stream and possibly coordinate barge-in directly (see Section 12.2.5 for more information). Finally, the TEARDOWN request-response is straightforward - the client indicates the session to teardown via the Session header field and the server responds with a status of 200 OK. The MRCP client can also use a DESCRIBE message to discover the capabilities of the media resource server in much the same way as MRCPv2 uses the SIP OPTIONS message (see Section 5.5.1). The response to the DESCRIBE message includes SDP describing the media characteristics supported for the given RTSP URI.

Once the session is initiated via the SETUP request, MRCP messages may be sent via the RTSP ANNOUNCE method. The MRCP request message is tunnelled in the body of the ANNOUNCE request and the MRCP response message is tunnelled in the ANNOUNCE response. For MRCP events, an ANNOUNCE message is sent from the media resource server to MRCP client with the event message contained in the ANNOUNCE message body. The message body of the ANNOUNCE response in this case is empty. Figure A.2 illustrates the MRCP SPEAK mechanism assuming the RTSP SETUP message has already completed.

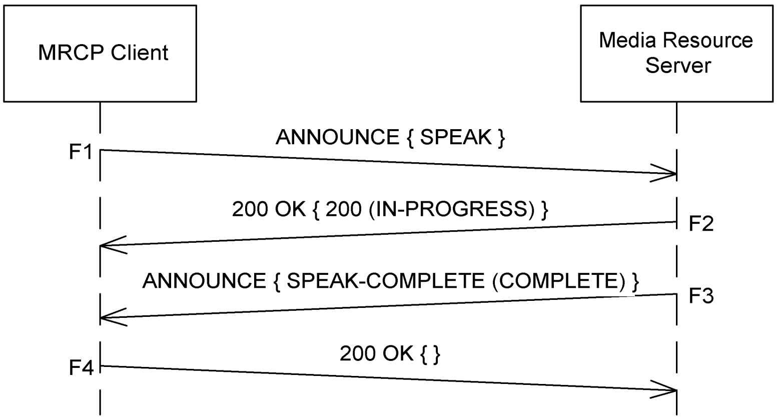

Figure A.2 MRCPv1 SPEAK example. MRCP messages are contained in the body of RTSP ANNOUNCE messages denoted by braces.

F1 (client → server):

```
ANNOUNCE rtsp://media.server.com/media/synthesizer RTSP/1.0
CSeq: 2
Session: 12345678
Content-Type: application/mrcp
Content-Length: 257

SPEAK 10000 MRCP/1.0
Content-Type: application/ssml+xml
Content-Length: 176

<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis">
   When I grow up, I want to be a protocol in my own right.
</speak>
```

F2 (server → client):

```
RTSP/1.0 200 OK
CSeq: 2
Session: 12345678
Content-Type: application/mrcp
Content-Length: 30

MRCP/1.0 10000 200 IN-PROGRESS
```

F3 (server → client):

```
ANNOUNCE rtsp://media.server.com/media/synthesizer RTSP/1.0
CSeq: 3
Session: 12345678
Content-Type: application/mrcp
Content-Length: 38

SPEAK-COMPLETE 10000 COMPLETE MRCP/1.0
```

F4 (client → server):

```
RTSP/1.0 200 OK
CSeq: 3
Session: 12345678
```

A.3 General Protocol Details

The astute reader may have noticed some subtle changes in the formatting for the start line of the MRCPv1 request, response line and event messages in comparison to MRCPv2. The differences between the two come about from the placement of the protocol version and the inclusion of a message length field in MRCPv2. See Table A.1 for a summary.

Table A.1 Comparison of start lines in MRCPv1 and MRCPv2 messages

Message type	Syntax
v1 request	`method-name request-id MRCP/1.0`
v2 request	`MRCP/2.0 msg-length method-name request-id`
v1 response	`MRCP/1.0 request-id status-code request-state`
v2 response	`MRCP/2.0 msg-length request-id status-code request-state`
v1 event	`event-name request-id request-state MRCP/1.0`
v2 event	`MRCP/2.0 msg-length event-name request-id request-state`

The status codes appearing in MRCPv1 and MRCPv2 responses are similarly defined. MRCPv2 adds a 410 error code for sequence number errors in the request and a set of 5xx error codes to indicate an error with the server. The MRCPv1 general header fields are a subset of MRCPv2s and are listed in Table A.2.

MRCPv2 adds several new general headers to the protocol (see Chapter 7 for more information):

- Channel-Identifier;
- Accept;
- Fetch-Timeout;
- Set-Cookie;
- Set-Cookie2;
- Vendor-Specific.

Table A.2 General header fields supported in MRCPv1

Header name	SET-PARAMS / GET-PARAMS
Active-Request-Id-List	No
Content-Base	No
Content-Encoding	No
Content-ID	No
Content-Length	No
Content-Location	No
Content-Type	No
Proxy-Sync-Id	No
Accept-Charset	No
Cache-Control	Yes
Logging-Tag	Yes

Note that unlike MRCPv2, MRCPv1 has no need for a Channel-Identifier header field since the logical MRCPv1 channel is bound to the RTSP session (identified by the RTSP Session header field) it is being tunnelled through. If a speech recogniser and speech synthesiser are used in the same session, the MRCP messages are differentiated by the RTSP URI.

A.4 Speech Synthesiser Resource

The speech synthesiser resource operates in much the same fashion in MRCPv1 as it does in MRCPv2. There is no concept of a basicsynth resource type, however. Tables A.3 + A.4 summarise the methods and events supported by the MRCPv1 synthesiser resource.

The methods and events are the same as MRCPv2 except that MRCPv2 adds the `DEFINE-LEXICON` method (see Section 12.2.7). The `SPEECH-MARKER` event is used in MRCPv1 only for reporting marks encountered in SSML and does not contain timestamp information. The resource-specific header fields in MRCPv1 are a subset of those in MRCPv2 and are listed in Table A.5.

Table A.3 Request messages supported by the MRCPv1 speech synthesiser resource

Message name	Description
`SPEAK`	Provides the speech synthesiser with input to synthesise and initiates speech synthesis and audio streaming.
`PAUSE`	Pauses speech synthesis.
`RESUME`	Resumes a paused speech synthesiser.
`STOP`	Stops speech synthesis.
`BARGE-IN-OCCURRED`	Communicates a barge-in event to the speech synthesiser.
`CONTROL`	Modifies the ongoing speech synthesis action.

Table A.4 Event messages supported by the MRCPv1 speech synthesiser resource

Message name	Description
`SPEECH-MARKER`	Generated when the speech synthesiser encounters a `<mark>` element.
`SPEAK-COMPLETE`	Generated when the corresponding `SPEAK` request completes.

MRCPv2 adds several additional header fields (see Section 12.4):

- `Completion-Reason`;
- `Audio-Fetch-Hint`;
- `Load-Lexicon`;
- `Lexicon-Search-Order`.

Table A.5 Resource-specific header fields for the MRCPv1 speech synthesiser resource

Header name	Request	Response	Event	SET-PARAMS / GET-PARAMS
Completion-Cause	-	-	SPEAK-COMPLETE	No
Failed-URI	-	SPEAK	SPEAK-COMPLETE	No
Failed-URI-Cause	-	SPEAK	SPEAK-COMPLETE	No
Speech-Marker	-	-	SPEECH-MARKER	No
Voice-Gender Voice-Age Voice-Variant Voice-Name	SPEAK CONTROL	-	-	Yes
Prosody-Pitch Prosody-Contour Prosody-Range Prosody-Rate Prosody-Duration Prosody-Volume	SPEAK CONTROL	-	-	Yes
Speaker-Profile	SPEAK	-	-	Yes
Speech-Language	SPEAK	-	-	Yes
Kill-On-Barge-In	SPEAK	-	-	Yes
Fetch-Hint	SPEAK	-	-	Yes
Jump-Target[a]	SPEAK CONTROL	-	-	No
Speak-Restart	-	CONTROL	-	No
Speak-Length	SPEAK CONTROL	-	-	No

[a] This header is renamed to Jump-Size in MRCPv2.

A.5 Speech Recogniser Resource

The speech recogniser resource operates in much the same fashion in MRCPv1 as it does in MRCPv2. There is no concept of a dtmfrecog resource versus a speechrecog resource. Further, the speech recogniser resource does not support hotword recognition or voice enrolled grammars, and does not support the concept of queuing recognition requests. The methods and events for the MRCPv1 speech recogniser are summarised in Tables A.6 and A.7.

The methods and events defined in MRCPv1 are essentially the same as MRCPv2 except that MRCPv2 adds the `INTERPRET` method and corresponding `INTERPRETATION-COMPLETE` event (see Section 13.2.6). In terms of header fields, MRCPv1 supports a subset of those specified in MRCPv2 - see Table A.8.

Table A.6 Request messages supported by the MRCPv1 speech recogniser resource

Message name	Description
`RECOGNIZE`	Requests that the recogniser resource commence recognition and specifies one or more grammars to activate.
`DEFINE-GRAMMAR`	Provides one or more grammars to the recogniser resource and requests that it access, fetch and compile the grammars as needed in preparation for recognition.
`RECOGNITION-START-TIMERS`[a]	Starts the no input timer.
`GET-RESULT`	Retrieves recognition results for a completed recognition. This method can be used by the MRCP client to specify a different confidence level or *N*-best list length to retrieve different result alternatives, for example.
`STOP`	Stops an ongoing recognition.

[a] This request is called `START-INPUT-TIMERS` in MRCPv2.

Table A.7 Event messages supported by the MRCPv1 speech recogniser resource

Message name	Description
`START-OF-SPEECH`[a]	Generated when the recogniser resource first detects speech or DTMF.
`RECOGNITION-COMPLETE`	Generated when the recognition request completes.

[a] This event is called `START-OF-INPUT` in MRCPv2.

MRCPv2 adds the following new header fields (Section 13.5):

- `Completion-Reason`;
- `Input-Waveform-URI`;

- `Input-Type`;
- `Media-Type`;
- `Ver-Buffer-Utterance`;
- `Recognition-Mode`;
- `Cancel-If-Queue`;
- `Hotword-Min-Duration`;
- `Hotword-Max-Duration`;
- `Interpret-Text`;
- `DTMF-Buffer-Time`;
- `Clear-DTMF-Buffer`;
- `Early-No-Match`.

Table A.8 Resource-specific header fields for the MRCPv1 speech recogniser resource

Header name	Request	Response	Event	SET-PARAMS / GET-PARAMS
Completion-Cause	-	RECOGNIZE DEFINE-GRAMMAR	RECOGNITION-COMPLETE	No
Failed-URI	-	RECOGNIZE DEFINE-GRAMMAR	RECOGNITION-COMPLETE	No
Failed-URI-Cause	-	RECOGNIZE DEFINE-GRAMMAR	RECOGNITION-COMPLETE	No
Confidence-Threshold	RECOGNIZE GET-RESULT	-	-	Yes
Sensitivity-Level	RECOGNIZE	-	-	Yes
Speed-Vs-Accuracy	RECOGNIZE	-	-	Yes
N-Best-List-Length	RECOGNIZE GET-RESULT	-	-	Yes
No-Input-Timeout	RECOGNIZE	-	-	Yes
Recognition-Timeout	RECOGNIZE	-	-	Yes
Speech-Complete-Timeout	RECOGNIZE	-	-	Yes
Speech-Incomplete-Timeout	RECOGNIZE	-	-	Yes
DTMF-Interdigit-Timeout	RECOGNIZE	-	-	Yes
DTMF-Term-Timeout	RECOGNIZE	-	-	Yes
DTMF-Term-Char	RECOGNIZE	-	-	Yes
Save-Waveform	RECOGNIZE	-	-	Yes
Waveform-URL[a]	-	-	RECOGNITION-COMPLETE	No
Recognizer-Start-Timers[b]	RECOGNIZE	-	-	No
Speech-Language	RECOGNIZE DEFINE-GRAMMAR	-	-	Yes
New-Audio-Channel	RECOGNIZE	-	-	No
Recognizer-Context-Block	-	-	-	Yes
Fetch-Timeout	RECOGNIZE DEFINE-GRAMMAR	-	-	Yes
Vendor-Specific	-	-	-	Yes

[a] This header field is called Waveform-URI in MRCPv2.

[b] This header field is called Start-Input-Timers in MRCPv2.