3GPP TR 24.880 V8.2.0 (2008-06)

Technical Report

3rd Generation Partnership Project;
Technical Specification Group Core Network and Terminals
Media server control using the IP Multimedia (IM)
Core Network (CN) subsystem;
Stage 3
(Release 8)

Keywords
UMTS, IMS, SIP, network, IP, multimedia,
stage 3

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document describes the implementation options and requirements for a media server control protocol to be used with the Mr and ISC interfaces.

This Technical Report includes the study of the following items:

- Define the requirements for a media server control protocol.
- Consider existing standards work that should be studied for the definition of a media server control protocol.
- Study and determine whether the media server control protocol should be carried in SIP or whether SIP should be used to setup a dedicated control interface.
- Study whether the AS directing the SIP session to the MRFC is always the same SIP-AS that should control the MRFC.
- Determine whether the media server control protocol should have a package naming and extension capability (similar to H.248 packages) to allow the support and registration of different media processing capabilities.
- Determine the SIP call flows and SDP capabilities associated with the media server control protocol and whether these SIP messages need to be passed through a S-CSCF proxy function or whether it is more efficient to have a direct AS-MRFC interface.

This Technical Report will be used to move into the specification phase for a media server control protocol.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] RFC 3261 (June 2002): "SIP: Session Initiation Protocol".

[3] RFC 4240 (December 2005): "Basic Network Media Services with SIP".

[4] draft-ietf-mediactrl-vxml: "SIP Interface to VoiceXML Media Services".

[5] W3C Recommendation (March 2004): "Voice Extensible Markup Language (VoiceXML) Version 2.0", http://www.w3.org/TR/2004/REC-voicexml20-20040316/.

[6] W3C Proposed Recommendation (April 2007): "Voice Extensible Markup Language (VoiceXML) Version 2.1", http://www.w3.org/TR/2007/PR-voicexml21-20070425/.

[7] W3C Working Draft (January 2007): "Voice Browser Call Control: CCXML Version 1.0", http://www.w3.org/TR/2007/WD-ccxml-20070119/.

[8] RFC 2616 (June 1999): "Hypertext Transfer Protocol -- HTTP/1.1".

[9] draft-ietf-mediactrl-sip-control-framework: "A Control Framework for the Session Initiation Protocol (SIP)".

[10] draft-mcglashan-mscp: "Media Server Control Protocol (MSCP)".

[11] RFC 4140 (September 2005): "TCP-Based Media Transport in the Session Description Protocol (SDP)".

[12] draft-ietf-simple-message-sessions: "The Message Session Relay Protocol".

[13] draft-ietf-speechsc-mrcpv2: "Media Resource Control Protocol Version 2 (MRCPv2)".

[14] draft-saleem-msml: "Media Server Markup Language (MSML)".

[15] RFC 4722 (Noivember, 2006) : "Media Server Control Markup Language (MSCML) and Protocol".

[16] draft-boulton-ivr-control-package: "A Basic Interactive Voice Response (IVR) Control Package for the Session Initiation Protocol (SIP)".

[17] draft-boulton-ivr-vxml-control-package: "A VoiceXML Interactive Voice Response (IVR) Control Package for the Session Initiation Protocol (SIP)".

[18] draft-boulton-conference-control-package: "A Conference Control Package for the Session Initiation Protocol (SIP)".

[19] RFC 4353 (February, 2006): "A Framework for Conferencing with the Session Initiation Protocol (SIP)"

[20] RFC 2976 (October, 2000): "The SIP INFO Method".

[21] International Packet Communications Consortium: "IPCC Reference Architecture V2" (June 2002).

[22] draft-ietf-mediactrl-requirements: "Media Server Control Protocol Requirements".

[23] draft-ietf-xcon-common-data-model: "A Common Conference Information Data Model for Centralized Conferencing (XCON)"

[24] 3GPP TS 24.147: "Conferencing using the IP Multimedia (IM) Core Network (CN) subsystem"

[25] RFC 4582 (November 2006): "The Binary Floor Control Protocol (BFCP)"

[26] RFC 4575 (August 2006): "A Session Initiation Protocol (SIP) Event Package for Conference State"

[27] RFC 4566 (July 2006): "Session Description Protocol (SDP)"

[28] RFC 3264 (June 02): "An Offer/Answer Model with the Session Description Protocol (SDP)"

[29] W3C Working Draft (February 2007): “State Chart XML (SCXML): State Machine Notation for Control Abstraction”, http://www.w3.org/TR/2007/WD-scxml-20070221/.

[30] RFC 2326 (April, 1998): “Real Time Streaming Protocol (RTSP)”.

[31] RFC 4245 (November 2005): "High-Level Requirements for Tightly Coupled SIP Conferencing".

[32] draft-ietf-xcon-framework: “A Framework and Data Model for Centralized Conferencing”.

[33] 3GPP TS 23.228: "IP multimedia subsystem; Stage 2".

[34] draft-boulton-mediactrl-mrb.txt: “Media Resource Brokering”

[35] 3GPP TS 23.218: "IP Multimedia (IM) session handling; IM call model; Stage 2".

[36] 3GPP TS 24.229: "IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3".

[37] 3GPP TS 23.333: "Multimedia Resource Function Controller (MRFC) – Multimedia Resource Function Processor (MRFP) Mp interface: Procedures Descriptions".

[38] 3GPP TS 24.247: "Messaging service using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3".

[39] 3GPP TS 32.240: "Telecommunication management; Charging management; Charging architecture and principles".

[40] 3GPP TS 32.260: "Telecommunication management; Charging management; IP Multimedia Subsystem (IMS) charging".

[41] 3GPP TS 32.210: "3G Security; Network Domain Security; IP network layer security".

[42] 3GPP TS 22.115: " Service aspects; Charging and billing".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TR 21.905 [1].

The following terms and definitions given in RFC 4353 [19] apply (unless otherwise specified):

Conference

Conference-URI

Conference-Aware participant

Conference policy

Focus

Mixer

Participant

Tightly Coupled Conference

Conference Notification Service

Conference policy server

The following terms and definitions given in draft-boulton-mediactrl-mrb-00.txt [34] apply (unless otherwise specified):

MRB

Query MRB

In-Line MRB

The following terms and definitions given in TS 32.210 [41] apply (unless otherwise specified):

Security Domain

Confidentiality

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TR 21.905 [1].

AS Application Server

MRFC	Multimedia Resource Function Controller
MRFP	Multimedia Resource Function Processor
S-CSCF	Serving CSCF

4 Media server control protocol study items

4.1 Introduction

The present section lists open topics that require study and decisions before the requirements for a media server control protocol can be finalized.

4.2 Controlling of Media Server vs controlling of Media Resource Function

Different from the Mp interface which uses a master-slave control protocol, the Mr interface between AS and MRFC is based on SIP which is a client-server protocol.

MRFC provides media control functions by accepting requests from AS, performing media control functions and reporting results by returning responses or sending notifications.

The connection relationship between AS and MRFC could be 1:1, 1: N, N: 1 and N: N.

The main functions required of MRFC media server control are:

a) To provide instructions to the MRFC on media operations to perform and the media sources and destinations to be used.

b) To provide information on resources required or to be reserved so that the MRFC can select and use an appropriate MRFP. Information on resources required may include geographic, QoS, SLA and priority information.

c) To provide information for the charging of the above operations and resource usage.

Most of the industry experience and the protocols referenced in the present document is based on the functions described in (a) above.

There is also ongoing work in the industry on addressing point (b) above via media resource brokers, which is further described in section 5.6. Note there are similarities in the functions of an MRB (in selecting an MRFC) and an MRFC (in selected an MRFP) in that the information required for the selection of the MRFP will be a subset of the information required for the selection of the MRFC.

These functions are fairly independent and separate protocol elements may be required for each. It should also be possible to standardize one of these functions independently of the others.

4.3 Choice of the mechanism and transport channel for media server control

4.3.1 Delegation model

The delegation model is motivated by the notion that that the interface between the MRFC and an AS is a high level interface where the MRFC is a network entity to which an AS delegates execution of media behavior.

The interface is high level since the AS sends a script describing what media behavior should be performed, not how it should be performed in terms of low-level media operations. The script describes the media behavior in terms of a flow of functions (play prompt, collect DTMF, add participant to conference, etc) and control logic for managing and adjusting the flow (e.g. adjusting for behavior in case of media operation failures), fetching additional scripts and resources, and reporting intermediate data.

The MRFC contains script engines which executes these scripts. The engine maintains the state of script execution and therefore the state of the media behavior execution. The engine's execution environment contains components to manage relationships with other components, including the low-level media processors. Consequently, when an AS 'delegates' execution of media behavior to a MRFC, it means the MRFC has an execution state which is independent of the AS's state – the MRFC not the AS manages the execution state of the media behavior. The controller instructs the MRFC which script to run, but the MRFC manages execution of the script itself.

In terms of architecture, this model uses the existing MRFC reference points, together with one additional reference point – the Sr reference point. Figure 4.3.1.1 shows an MRFC with this reference point.

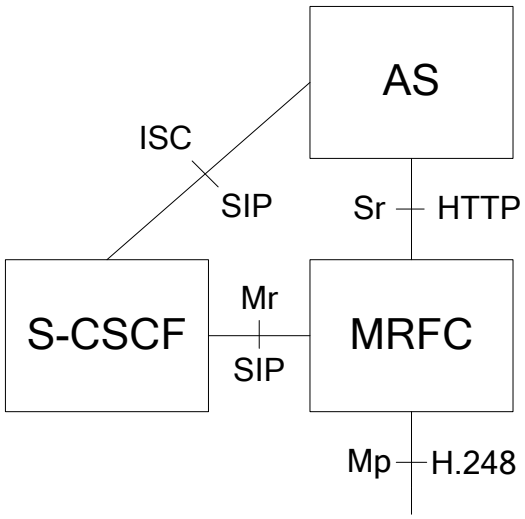


Figure 4.3.1.1 MRFC reference points: Sr, Mr and Mp

Using the ISC interface, an AS establishes a SIP [2] dialog to an MRFC (via a S-CSCF and Mr interface). The SIP INVITE request URI shall contain sufficient information to allow the MRFC to identify the script to execute; it may also provide additional parameters for the script. For example, using the user part to indicate a script pre-defined on the MRFC:

```
INVITE sip:myservice@mrf.example.com  SIP/2.0
```

where "myservice" is predefined with a script on the MRFC, or specifying a script URI as a parameter:

```
INVITE sip:dialog@mrf.example.com;voicexml=http://server.example.com/script.vxml    SIP/2.0
```

where a VoiceXML script is specified as the value of the parameter "voicexml". IETF Informational RFC 4240 [3] and Working Drafts draft-ietf-mediactrl-vxml [4] provide details on this mechanism.

The Sr interface is used by the MRFC to fetch the script and related resources. Once these have been fetched, the script is executed by the MRFC. Depending on the contents of the script, its execution may involve sending data and fetching additional scripts and resources over the Sr interface. The interaction is terminated when a SIP BYE is sent; the AS can send a BYE to terminate script execution at any time, and the MRFC sends a BYE when execution of the script terminates.

The content of the scripts is dependent on the media behavior which the MRFC needs to execute. W3C has already done extensive work on defining scripting for use in the delegation model. VoiceXML [5] provides a scripting language for interactive media functions; VoiceXML [5] is motivated in Section 6.2.1. CCXML [7] provides a scripting language for conferencing, dialog invocation and outbound dialing; CCXML [7] is motivated in Section 6.2.2. SCXML [29] is a generic event-driven state machine language which can be extended with support for dialog and conferencing functionality. SCXML is motivated in subclause 6.2.3,

In several scenarios, scripts executed by the MRFC may request to perform actions which may not be allowed on MRFC. Such actions may include, but are not limited to, outgoing call establishment and call transfer (since the MRFC description is not clear whether these are permitted MRFC functions). For such scenarios a mechanism should be defined to deliver the action information from MRFC to AS and then performing the action by AS. Such mechanism may utilize existing interfaces between MRFC and AS (i.e. ISC and Mr) or new ones (Sr or Cr – see below).

RFC 4240 [3] provides fundamental technique for the delegation model, but it alone is insufficient for the range IVR and conferencing functions of the MRFC. RFC4240 is necessary since the delegation model uses the content of the Request-URI in an INVITE to identify and invoke media services. Each of these services imposes different requirements in terms of MRFC script engine complexity. The announcement service requires a simple engine which uses the Sr interface to fetch media resources. Likewise, the conference service requires a simple engine for simple conferencing. The VoiceXML dialog service requires a more complex script engine, but VoiceXML is well understood. Moreover, the description of the VoiceXML service in RFC4240 is incomplete and raises a number of issues. The description of this service in draft-ietf-mediactrl-vxml [4] (which builds on RFC4240) addresses most of these issues (whether an MRFC can initiate outbound calls is still outstanding). If the deficiencies in RFC4240 can be addressed in conjunction with other specifications, RFC4240 can provide a straightforward approach for identifying and invoking simple announcement and conferencing services as well as complex IVR services.

4.3.1.1 New reference point: Sr

The delegation model requires a new MRFC reference point, "Sr".

The 3GPP SA2 group would have to be consulted for the creation of this new reference point.

The Sr interface enables the MRFC to fetch documents (scripts and other resources) from an entity on the application plane.

The entity can provide these documents either from local storage or generated at runtime. The entity may be an AS if the AS supports the protocol requirements below.

The Sr interface is asymmetrical: fetch requests are only initiated by the MRFC – the application plane entity can only respond to requests.

HTTP [8] is an asymmetrical protocol which is extensively deployed for document fetching. HTTP also provides a caching model which permits fetches optimization and can thereby reduce traffic on the network. For example, documents may be fetched only when they have expired in the local cache; and fetching can be configured so that documents are not fetched at all if there is an unexpired version in the local cache.

The Sr interface shall support the HTTP [8] protocol (including full caching capabilities). Specifically, the MRFC shall support the HTTP client role and the application plane entity shall support the HTTP server role. The Sr interface should support HTTPS (where IMS network topology requires a secure connection is required). The Sr interface may support other protocols with an asymmetrical request-response model.

4.3.1.2 Mid-call XML support

The delegation model could provide mid-call XML support by extending the Sr interface so that it symmetrical.

Mid-call XML is a technique to allow the intelligence of the service to reside on an AS which asynchronously sends commands as XML fragments to the MRFC thereby driving the behavior of IVR and conferencing services.

In the delegation model, these XML fragments could be delivered to MRFC script engines over the Mr interface using SIP INFO. However, an approach using SIP INFO to pass control data would need to address the problems raised in Section 4.3.2.

An alternative approach for the delegation model is that the XML fragments are delivered to MRFC script engines over the Sr interface. This would require that the Sr interface becomes symmetrical: just as the MRFC can initiate HTTP requests to the AS, so the AS would also be able to initiate HTTP requests towards the MRFC.

Script engine would then need to support receiving XML fragments in an HTTP request. In VoiceXML 2.0/2.1, there is no support: while a VoiceXML MRFC can initiate HTTP requests to the AS and receive XML in return, it cannot accept HTTP requests itself. VoiceXML 3.0 is expected to support asynchronous events, potentially with XML payloads. With CCXML, however, there is support for receiving HTTP requests (its Basic HTTP input-output processor is bi-directional), and these requests could contain XML fragments which provide instructions for advanced conference control.

4.3.1.3 Example

The diagram in Figure 4.3.1.3.1 shows a simple delegation case where the MRFC uses a VoiceXML script to prompt the user for digits and return them to the AS.

Note that the SIP signaling between the CSCF and the AS, and between the CSCF and the UE, has been omitted for the sake of clarity.

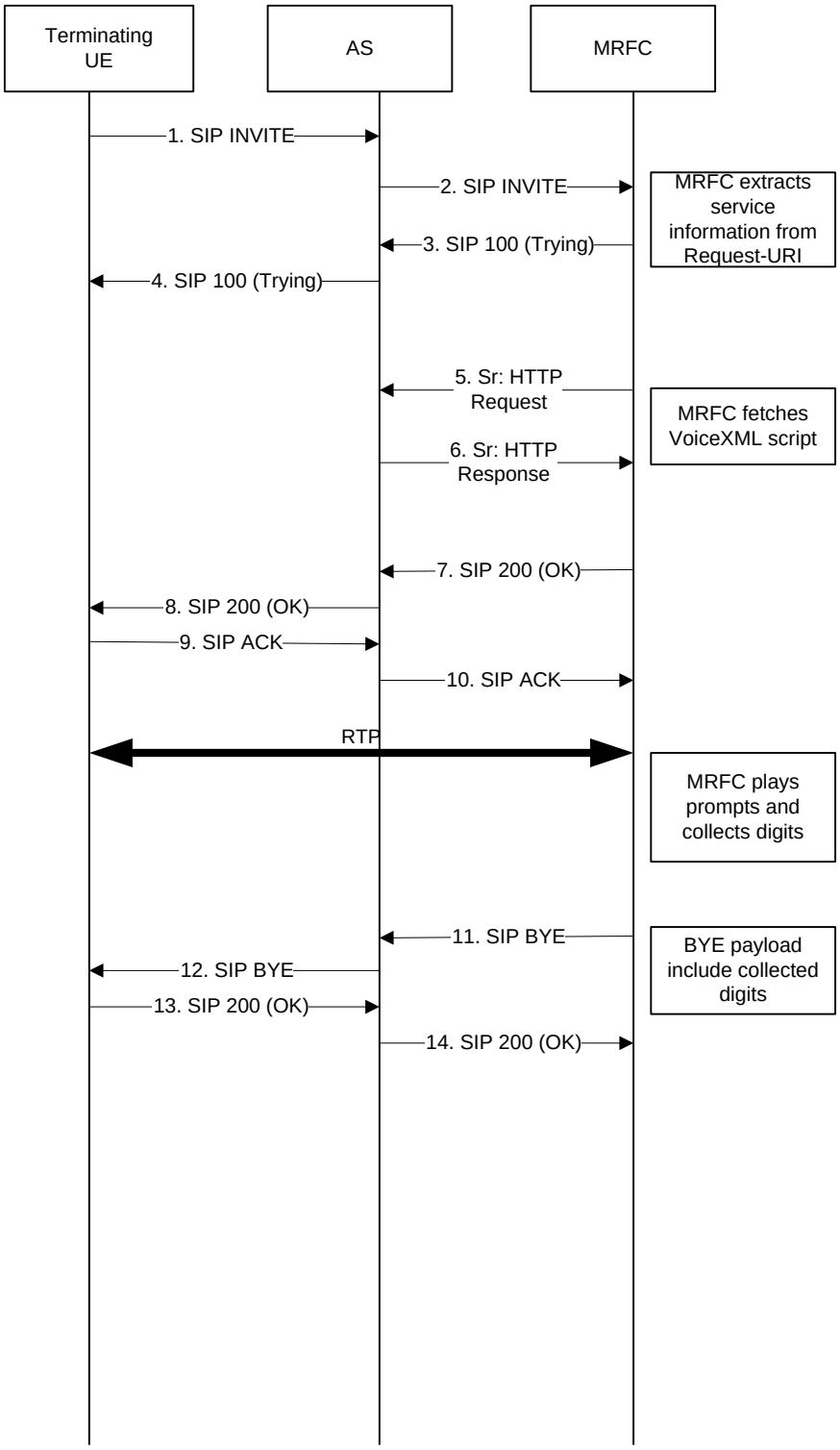


Figure 4.3.1.3.1: Delegation Model with simple prompt and collect call flow

In step 2, the MRFC extracts the VoiceXML script URI from the SIP INVITE Request-URI. See Section 4.3.1 for examples of Request-URI's.

In steps 5 and 6, the MRFC fetches the VoiceXML document from the AS using HTTP over the Sr interface. These steps would be repeated if additional resources were required; for example, prompt files. Note that these steps could be eliminated if the VoiceXML document and resources were already cached on the MRFC.

Once the RTP channel is established, the MRFC executes the VoiceXML script playing any prompts and waiting for digits from the user. Once the digits are collected, the MRFC terminates the SIP dialog in step 11 and return the collected digits in the SIP BYE body. Alternatively, the MRFC could have sent the result to the AS using HTTP over the Sr interface.

4.3.1.4 Properties

As a high-level interface, the delegation model is clearly distinguished from, and complements, the low-level H.248 model on the Mp interface. Application developers can use a high-level model – familiar to web application developers – where they script their media interaction and delegate it to the MRFC, or they can develop using a low-level model – familiar to the API developers - where they use a TCP connection to send detailed instructions to the MRFP and then manage its state themselves. In the delegation model, the media behavior is defined in a script at the application service layer, the control layer (MRFC) which executes the script and manages media flow, and the media layer (MRFP) which actually carries out the media functions specified in the script. In a low-level model, the service and control layers are combined in a hybrid AS/MRFC.

With the delegation model, the AS can choose how much control to delegate to the MRFC. This depends on the content of the script and the behavior the script can execute before it needs to fetch a new script through the Sr interface. The AS can then exercise fine-grained (tight, low-level) or coarse-grained (loose, high-level) control and can modulate this within a session. Approaches which use a dedicated control channel typically require the AS to retain fine-grained control for the whole session.

The delegation model has been extensively tested and deployed as part the web infra-structure model where it has been demonstrated as highly suitable for distributed service architectures. By reusing a well-tried model, 3GPP can focus on definition of MRFC profiles.

The delegation model fits with existing MRF architecture with only the addition of one new reference point (which would be required by most alternative approaches if they explicitly recognized the need for an HTTP [8] fetching interface).

The Sr interface uses a well-known HTTP [8] protocol to fetch resources and provide responses/notifications.

The delegation model reduces the burden on the AS/CSCF to track the status, and interact with the MRFC, for the media part of interactive media, call and conferencing applications. This results in reduced network traffic with the MRFC since decisions about media flow are taken within the MRFC itself rather than passed up to the AS/CSCF for decision. For example, a single CCXML [7] or SCXML [29] script can be used to play announcement dialogs and to manage participants attending a conference, where a protocol approach will require multiple documents for creating the conference, playing dialogs, and adding/removing conference participants. Furthermore, this can reduce the response time for media control management: i.e. since the MRFC manages the flow locally, there is no need to request the AS/CSCF (e.g. via SIP INFO on ISC/Mr or a dedicated control TCP channel) to make a decision and await a response.

Use of VoiceXML [5][6] and CCXML [7] support the core functions of the MRF and allows simple as well as complex interactive behavior defined in scripts. Existing VoiceXML and CCXML applications (e.g. voice mail, prepaid, portals, self-service applications) can be easily and rapidly adopted within a 3GPP IMS context without the need for application recoding. SCXML [29] together with a profile which defines call, dialog and conferencing functionality required for an MRFC, is an alternative to CCXML.

As W3C languages, VoiceXML, SCXML and CCXML are developed and supported by an official W3C working group. There is minimum dependency on IETF working drafts submitted by individuals.

The Mr and ISC interface are only used for call-related functions (call establishment, management and tear-down): it is not used to transmit detailed media control messages to the MRFC or to establish dedicated control channels with the AS.

The delegation model facilitates different entities on the application layer to play different roles with respect to the MRFC. For example, a ‘gateway’ AS may initiate the sessions via the Mr interface, while others can receive HTTP requests and notifications via the Sr interface. Protocol-based approaches typically assume that the same AS which initiates the media session also interacts with the MRFC during the session.

4.3.2 Protocol model with dedicated control channel

The protocol model is motivated by the notion that the interface between the MRFC and AS is a high level interface where the AS uses a transport channel to send media control messages to the MRFC. The MRFC executes the messages and sends responses and notifications back through the transport channel.

The protocol model could use either the ISC and Mr interfaces (e.g. messages in SIP INFO) or a new interface (Cr – see below) with a dedicated transport channel to transmit media control messages. The majority of deployed approaches which follow the protocol model use mechanisms that include carrying commands in a SIP INFO method. This has been an appropriate short term solution during the evolution of SIP [2] and has facilitated early deployments but does not provide a roadmap for future success in the standards arenas. The following outlines some of the reasons that using techniques such as SIP INFO are not considered appropriate:

- SIP INFO was created 'to carry session control information along the SIP signaling path. It merely sends optional application information, generally related to the session'. Examples of SIP INFO method-use included in the draft are carrying mid-call PSTN signaling messages between PSTN gateways and DTMF digits. This mechanism in not suited or ideally appropriate for carrying information such as media control messages. For this reason alone any mechanism that uses SIP INFO will never be accepted as an industry standard within the IETF.

- The default protocol for SIP is the Unicast Datagram Protocol (UDP). Using SIP and UDP for transfer of media server commands is unreliable and also inherits problems with large packet size. Media server control messages should always be sent over reliable, congestion safe protocols.

- When using a mechanism like SIP INFO, it is possible that any number of intermediaries can insert themselves into the signaling path, either as a record routing proxy or 'Back-to-Back User Agent (B2BUA), This would result in media server control messages being carried in SIP INFO across any number of SIP intermediaries, which is not ideal or efficient in large networks. There is also the overhead of using a full SIP message with all its mandatory headers and transaction timers which can impact performance dramatically.

- The core SIP specification, RFC3261 [2], contains rules when un-reliable transport protocols such as UDP are used. If a packet reaches the Maximum Transmission Unit (MTU), the transport protocol is upgraded to a reliable form such as TCP. This type of operation is not ideal when constantly dealing with large payloads which are present in a media server control messages.

Identifying such problems – many arising from practical deployment experience - indicates that an alternative mechanism is required for MRFC control that not only leverages the benefits of SIP but also dispels the previously identified problem areas.

The alternative, as described in the SIP Control Framework [9] - under discussion within the IETF mediactrl working group - is to carry media control messages over a dedicated control channel (SIP Control Framework [9], MSCP [10] - note that while MSCP version 1 defined its own control channel, MSCP version 2 uses the Control Framework).

In the Control Framework SIP is used for its intended purpose – as a rendezvous protocol for negotiating a media session using the Session Description Protocol (SDP). Unlike SIP dialogs with UEs where the SDP are used to establish RTP media streams between the MRF and UE, the approach leverages COMEDIA (RFC4145) [11] so that the SDPs described the establishment of a TCP (or SCTP) channel. The COMEDIA [11] approach is well established and used in the Message Session Relay Protocol (MSRP) [12] which initiates IM media sessions (MSN, Yahoo style chat interactions as apposed to 'one-hit' SMS style messages), as well as in Media Resource Control Protocol (MRCP) [13] which establishes a TCP channel to transport control messages to/from speech recognition and speech synthesis media processors. Thus, MRFC messages are exchanged over a direct (peer-to-peer) connection, using a reliable protocol, where the protocol has been initiated using SIP. This addresses the previously identified problems that arose when using SIP INFO:

- SIP INFO method is not used as the approach defines its own message primitives that are passed across the dedicated control channel. This eradicates the inappropriate use of the SIP INFO message.

- The approach only uses reliable connection orientated protocols such as TCP (or SCTP) so messages passing across the control channel are sent reliably.

- As the control channel connection is peer-to-peer it doesn't matter how many intermediaries the SIP signaling traverses. The media control messages will always pass directly. These messages are also extremely light-weight and do not suffer from complicated transaction models.

- As the dedicated control channel is created using a reliable protocol such as TCP, and SIP is not used to pass interactions, this mechanism does not suffer from the MTU upgrading define in RFC 3261 [2].

The Control Framework approach itself does not define the content of messages transported by the dedicated control channel: its development was motivated by the media control scenario, but it is expected that the Control Framework could be used in a wide variety of application scenarios in the future. Instead the framework defines a mechanism that provides strict requirements on how the Control Framework can be used. Techniques similar to the SIP Event Framework (RFC 3265) are used when creating extensions to the Control Framework. The Control Framework introduces the concept of 'Control Packages'. Control Package authors are provided a strict set of rules that shall be followed to use the Control Framework.

The use of packages in the control framework is motivated by the fact that media server control is a complex topic area with a wide range of potential functionality encompassing many varying technologies. Within IMS, the functionality of the MRF is a moving target; while interactive media (play prompt, prompt and collect, etc) as well as conferencing are core functionalities, the ever expanding IMS world also makes it highly likely that technologies will advance in the coming years; MRFs with new functionalities as well as MRFs which combine interactive media and/or conferencing with new ones. It is for this reason that any solution for MRFC needs to be modular in nature and highly extensible. This then allows infra-structure providers and application developers to select only the relevant subset of technology required instead of dealing with enormous, monolithic command sets that are quite often redundant. For this reason, the media control functionality shall be organized into packages.

Various IETF working drafts proposals on media server protocol have started to move from the monolithic commands sets towards functionality organized into packages; for example, MSML [14] and MSCP [10]. MSCP [10] (version 2) uses the same packages as those being defined for the Control Framework:

- Basic Interactive Voice Response (IVR) Control Package [16]: This provides lightweight messages for simple IVR interactions. This control package uses parameterized dialog templates for playing announcement, prompt and collects and prompt and record IVR functions without the need to implement a full VoiceXML solution.

- VoiceXML Interactive Voice Response (IVR) Control Package [17]: This package extends the basic IVR control package with support for VoiceXML. Note that this package does not support VoiceXML's optional call transfer functionality.

- Conference Control Package [18]: This package allows for the creation, manipulation and termination of a conference mix. Users, explicitly represented by SIP dialog parameters, can be introduced, moved and removed from an existing conference mix.

Although still in early stages, these packages are starting to mature and provide a wide range of MRF functionality. It is expected over the coming period that both the Control framework and packages will mature. One of the next steps is a complimentary extension that provides video support to the appropriate control package and to enhance the Conference Package with support for conferencing. It is expected this document will be available in the very near future.

The use of VoiceXML [5][6] for IVR functionality, especially complex IVR functionality, is a shared feature in IETF informational RFCs and working drafts; for example, RFC4240 [3], MSCML [15], MSML [14], MSCP [10] and the VoiceXML control package [17] above.

VoiceXML scripts can be referenced (or included inline) as part of media control messages; for example, the message

```
<dialogstart src=" http://server.example.com/script.vxml"  type="application/voicexml+xml"/>
```

could be sent from the AS to the MRFC in order to initiate a VoiceXML dialog. Response and notifications about the dialog (dialogstarted, dialogexit, dialogerror, etc) are sent back over the control channel.

One consequent of using VoiceXML is that the VoiceXML scripts and its related resources need to be fetched from an entity on the application plane. The requirement still holds even if the initial VoiceXML script is specified inline in the media control message (as MSCP and the VoiceXML Control Package allow) since subsequent VoiceXML scripts as well as resources (such as grammars) may still need to be fetched. Furthermore, if any control package references resource using HTTP [8] URIs, then the MRFC shall support an interface which allows these resources to be fetched.

In terms of architecture, this model uses the existing MRFC reference points together with one additional reference point: a Cr reference point to directly transport media control messages between the AS and MRFC and to allow the MRFC to fetch resources. Figure 4.3.2.1 shows an MRFC with this reference point.

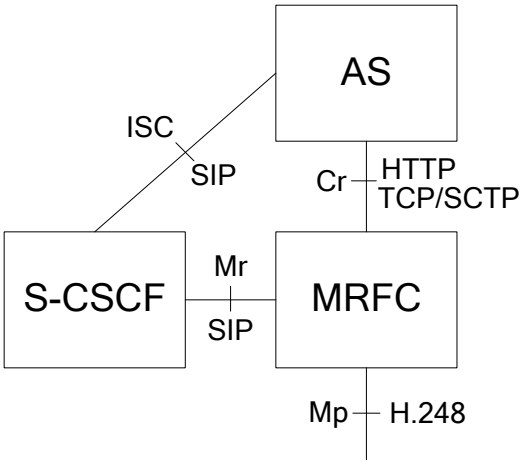


Figure 4.3.2.1: MRFC reference points: Cr, Mr and Mp

Note that the framework allows the AS to establish multiple dedicated control channels towards the MRFC; it could for example use one channel per MRFC, one channel per session, or other configurations suitable for High Availability deployments.

In the case of one channel per MRFC, it's required that the protocol used over the Cr interface is able to associate media control messages with the related SIP dialog(s) between the AS and the MRFC.

In the case of one control channel for one MRFC, in the protocol model with dedicated control channel, all the media control commands will go through one transport connection. Congestion may happen if the message traffic is high and the media control functionality will not be available if the underlying transport connection is down. So, it shall be possible for the media control protocol to use a transport layer protocol with high availability (e.g. SCTP) and load balancing to minimize the possibility of congestion.

The appropriate transport channel for the dedicated control channel should be specified depending on the availability requirements.

4.3.2.1 New reference point: Cr

The protocol model with dedicated control channel requires a new MRFC reference point, "Cr". The 3GPP SA2 group would have to be consulted for the creation of this new reference point.

Dedicated TCP/SCTP channels between the AS and MRFC flow over the Cr interface. Cr is using two types of TCP/SCTP channels: one dedicated for SIP control framework and the other for HTTP communication.

Media control packages are transmitted bi-directionality over the channels: either endpoint can send requests, responses and notifications depending on the package definitions.

The establishment and management of these channels shall follow the SIP Control Framework: i.e. using SIP over the Mr interface to establish the channel, and to negotiate control package support.

The Cr interface enables the MRFC to fetch documents (scripts and other resources) from an entity on the application plane.

The entity can provide these documents either from local storage or generated at runtime. The entity may be an AS if the AS supports the protocol requirements below.

The Cr interface's use for fetching documents is asymmetrical: fetch requests are only initiated by the MRFC – the application plane entity can only respond to requests.

HTTP [8] is an asymmetrical protocol which is extensively deployed for document fetching. HTTP also provides a caching model which permits fetches optimization and can thereby reduce traffic on the network. For example, documents may be fetched only when they have expired in the local cache; and fetching can be configured so that documents are not fetched at all if there is an unexpired version in the local cache.

The Cr interface shall support the HTTP [8] protocol (including full caching capabilities). Specifically, the MRFC shall support the HTTP client role and the application plane entity shall support the HTTP server role. The Cr interface

should support HTTPS (where IMS network topology requires a secure connection is required). The Cr interface may support other protocols with an asymmetrical request-response model.

4.3.2.2 Example

The diagram in Figure 4.3.2.2.1 shows a simple TCP control channel case where the MRFC uses a VoiceXML script to prompt the user for digits and return them to the AS.

Note that the SIP signaling between the CSCF and the AS, and between the CSCF and the UE, has been omitted for the sake of clarity.

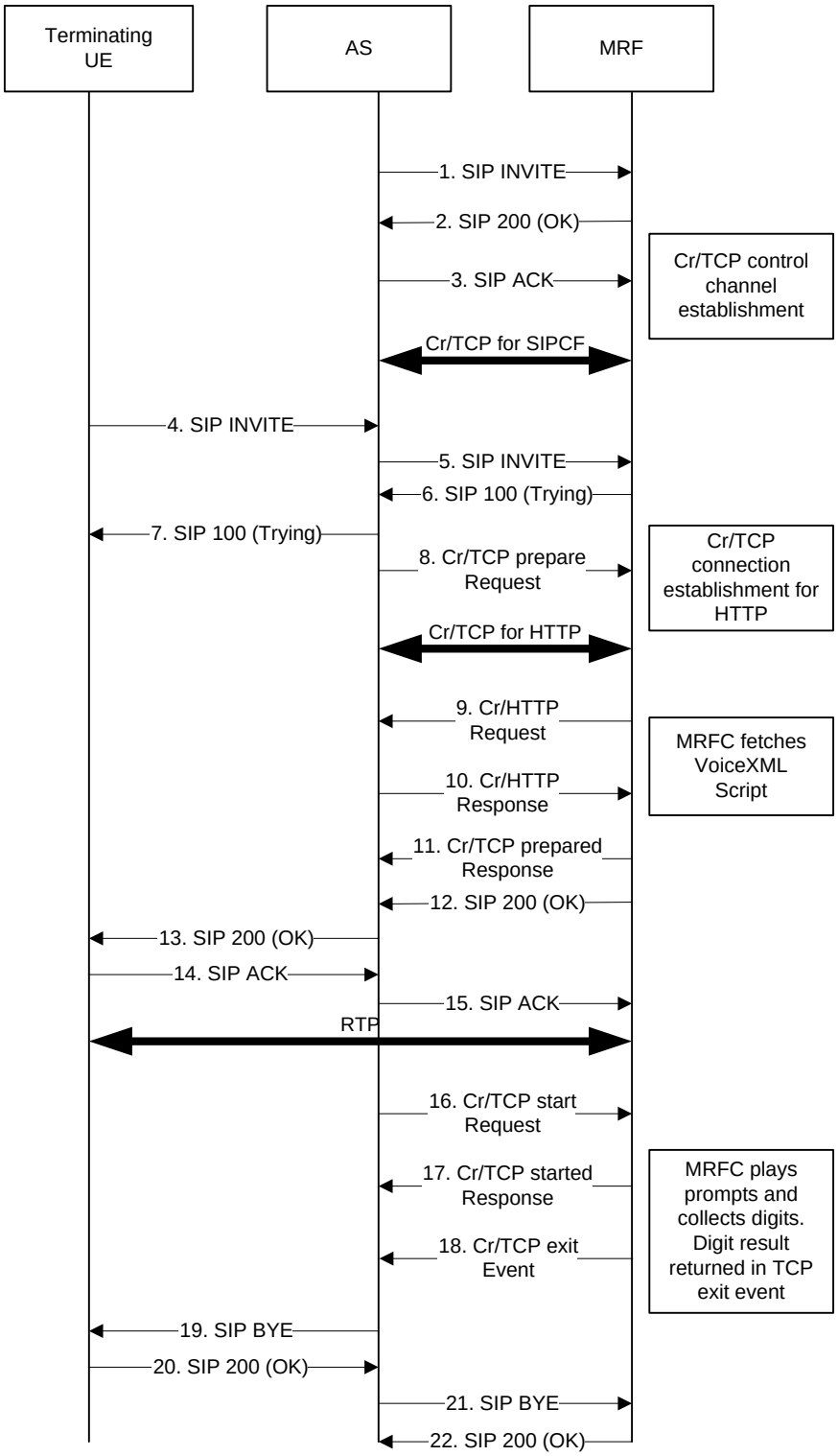


Figure 4.3.2.2.1: TCP Control Channel Model with simple prompt and collect call flow

In steps 1-3, the AS and MRFC establish a TCP control channel over the Cr interface. The same control channel can be used to control multiple calls.

During the UE call setup, the AS instructs the MRFC to prepare a VoiceXML dialog at step 8. Then MRFC establishes TCP connection to AS which is to be used for HTTP communication. The same HTTP connection can be used to perform multiple HTTP requests. In steps 9 and 10, the MRFC fetches the VoiceXML document from the AS using HTTP over the Cr interface. These steps would be repeated if additional resources were required; for example, prompt files. Note that these steps could be eliminated if the VoiceXML document and resources were already cached on the

MRFC. Once the script and resources are prepared, the MRFC sends a prepared response to the AS over the Cr control channel in step 11.

Once the RTP channel is established, the AS instructs the MRFC in step 16 to start executing the VoiceXML script: the MRFC plays any prompts and waits for digits from the user. Once the digits are collected, the MRFC returns the result to the AS in an exit event at step 18. The AS then terminates the SIP dialogs with the MRFC and UE in steps 19-22. Note that instead of terminating the call, the AS could have instructed the MRFC over the Cr/TCP channel to start another dialog, add the user to a conference, etc.

4.3.2.3 Properties

The protocol model uses a dedicated transport channel to transmit media control messages between the MRFC and AS. This avoids the problems described above with transmitting these messages over SIP INFO. The dedicated control channel in Control Framework has growing support within IETF.

The protocol model organizes media control messages into packages. This allows different MRFs to support different functionality package and, as described in the Control Framework [9], for an AS to determine which packages are supported by which ASs. Packages also facilitate future extensions to MRF functionality.

The protocol model's Cr interface shares many similarities with the Mp interface including use of TCP connections over which messages organized by functionality are transmitted. Refer to section 6.2.3 which describes the Mp interface in detail. The protocol model also provides an explicit mechanism for discovery and establishment of the control channel.

AS developers can use the protocol model within familiar API development environment which allows TCP connections to be created and XML messages transmitted over them. The state of media interaction is managed centrally within their application and they have full control over the MRFC since responses and notifications are returned over the control channel. At the same time, they can choose to delegate part of an IVR interaction to the MRFC by using the VoiceXML control package [17]: the MRFC would then locally manage the VoiceXML interaction while the AS retains global management (it receives notifications on key changes of dialog state – started, exited, etc – through the control channel).

The protocol model fits with the existing MRFC architecture with the addition of one new reference point, Cr. The Cr interface uses a well-known HTTP [8] protocol to fetch resources and is based on an emerging protocol with growing IETF support, and its setup is based on COMEDIA [11] which is well-established.

Use of VoiceXML [5][6] in control messages covers the IVR functions of the MRF and allows simple as well as complex interactive behavior to be defined in scripts. Existing VoiceXML applications (e.g. voice mail, prepaid, portals, self-service applications) can be easily and rapidly adopted within a 3GPP IMS context with minimal application recoding.

4.3.3 RFC 4240 (Netann) support

Some types of media processing can be driven entirely by the Request-URI in a SIP INVITE. These types of media processing support:

- playing an announcement and then disconnecting the bearer
- connecting the bearer to a simple conference
- invoking a VoiceXML interpreter on the bearer for IVR (with a side HTTP connection to an HTTP server with server-side scripting)

Mr needs to support RFC 4240 (Netann) in order to provide support for this type of media processing. RFC 4240 is well accepted in the industry.

4.3.4 Mid-call XML support

Some types of media processing require mid-call control between the AS and MRFC. These types of media processing support functions such as:

- advanced conferencing where the intelligence is in the AS and the AS passes commands asynchronously to the MRFC during the session

- IVR where the intelligence is in the AS and the AS passes commands asynchronously to the MRFC during the session

Mr needs to support an XML-based mid-call control scheme in order to provide support for this type of media processing. The XML can be carried in SIP INFO and/or in a long-lived, SIP-negotiated TCP/IP control channel. This type of SIP AS-MRFC interaction is well accepted in the industry.

RFC 4722 is an example of such a protocol. This informational RFC specifies the Media Server Control Markup Language (MSCML). The protocol uses concepts of Mid-Call XML support at its heart. More details on how RFC 4722 can be used for mid-call control are provided later in this document (see section 6.4.2).

4.3.5 Conferencing examples

Two conferencing examples are given here which show that different media control mechanisms may be appropriate for different use-cases.

The first example is a dial-in conference example (the requirements being those listed in RFC 4245 [31]). The reservation for this type of conference is typically done by an out of band mechanism and in advance of the actual conference time. The conference identification, which may be a URI with a pin number, is allocated by the reservation system. It is sent to all participants using email, IM, etc. The participants join using the conference identification. The conference identification must be routable enabling the allocation of a conference with free resources at the time when the conference actually runs.

The most important part of this application is implemented at the reservation time by the conferencing AS: to allocate an unique identifier for this reserved conference, to publish the conference's focus URI (and potentially a pin number) to all participant and finally to guarantee that necessary conference resources will be allocated at the time of the conference. We can easily imagine that the reservation step leads to create a conference policy as per [23] that includes a media conference policy (to typically reserve the conference resources).

In order to implement such a scenario where low level control is not required during the conference, the NETANN (RFC 4240 [3]) conferencing service in addition to media conference policy delegation is sufficient for the media control required. For more details on the media conference policy please refer to subclause 4.4.1.

The second example is of an enterprise's conferencing application where a human operator can control in real-time the execution of each conference. For simplicity we can imagine that the reservation mechanism is similar to the one described above. The fact that conference control is performed by an operator implies some more complexity:

- A participant can at any point in time ask for assistance for an operator, for instance to increase or decrease their volume gain.
- The conferencing AS must be aware of the action taken by each participant in order to inform the operator (via a web GUI for instance).
- The operator must have a way to control the execution of the conference like muting or un-muting a participant, ejecting a participant or adding a new participant.
- The operator (or a participant with a special role) can be asked to split a main conference into sub-conferences and merge them back afterwards.

Conversely to the previous example, important interactions need to take place between the conferencing AS and the MRFC/MRFP during the overall conference execution. These interactions can be commands initiated by the conferencing AS and corresponding responses from the MRFC/MRFP, or notifications coming from the MRFC/MRFP.

In this example the protocol model is more suited to provide support for these continuous interactions. In this model the media conference policy delegation described in subclause 4.4.1 can still be used.

4.4 AS and MRFC functional split for conferencing

This section is aimed to introduce the SIP tightly coupled conference and the collocated AS/MRFC model depicted in [24]. The terminology and concepts are re-used from the corresponding standard [19]. Please Note that the on-going 3GPP work described in [24] is based on a subset of [19].

The subsequent section will explore the decomposed AS/MRFC model depicted in [19] where the conference functionality is split over the conferencing application server (hereafter called AS) and the MRFC.

The XCON Centralized Conferencing Framework specified in draft-ietf-xcon-framework-07 [32] is compatible with the functional model presented hereafter. The subclause 4.4.1 introduces the role of AS as top-level focus, as described in [19], that handles the one-to-one relationship with the active conference object, while the low-level focus in MRFC is dedicated to control the media mixer(s).

A SIP tightly coupled model conference is an association of SIP user agents (i.e., conference participants) with a central point (i.e., a conference focus), where the focus has direct peer-wise relationships with the participants by maintaining a separate SIP dialog with each. The focus is a SIP user agent that has abilities to host SIP conferences including their creation, maintenance, and manipulation using SIP call control means (and potentially other non-SIP means). In this tightly coupled model depicted hereafter, the SIP conference graph is always a centralized star. The conference focus maintains the correlation among conference's dialogs internally.

As stated in [24] section 5.2.3 the functional split between the MRFC and the conferencing AS is out of scope, this section is focused to describe this model while the next section will depict the functional split.

The following figure depicts the main logical functions that are located at the AS/MRFC and MRFP levels.

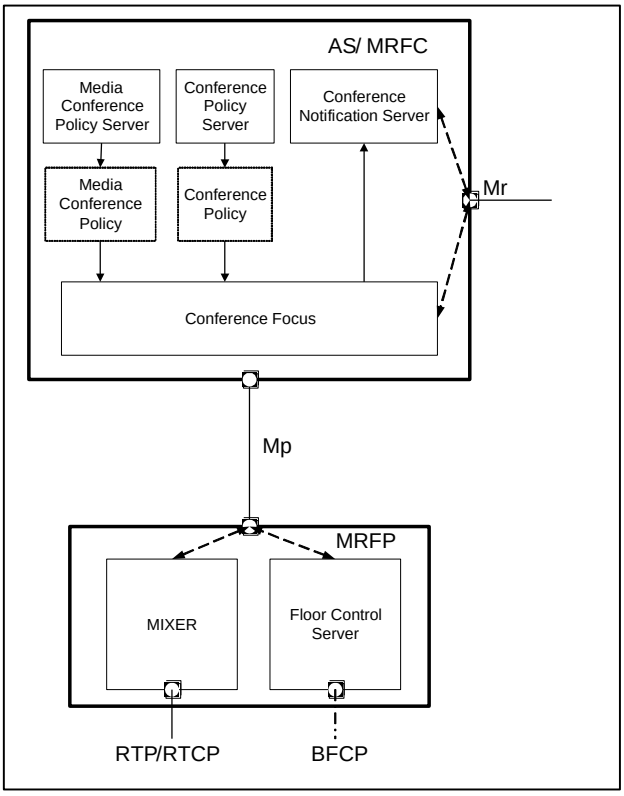


Figure 4.4.1: Conference logical functions spread over AS/MRFC and MRFP

As stated in [24] the conference focus, the conference policy server, media conference policy server and the notification server are collocated in the AS/MRFC.

For a given conference, the conference policy server is in charge to provide the conference policy, and the media conference policy server to provide the Media conference policy. The Conference focus is in charge to load these 2 conference policies at conference creation time and to govern the conference execution accordingly. These conference policies are XML based file defined in [23] (note that [23] defines the global data model where the 2 policies are combined). The conference focus informs the conference notification server on conference state changes, it is in charge to provide support for the conference notification service defines in [24] section 5.3.3.

The MRFP is connected to MRFC(s) through the Mp reference point; it hosts the Mixer function and the floor control server function as defined in [25]. The Mixer is connected to the UE through the RTP/RTCP protocols and the floor control server is connected to the floor control client (hosted by the UE) through the Binary Floor Control Protocol as

defined in [25]. The Mp interface is intended to carry the commands provided by the conference focus to the mixer and to send back events from the mixer, in addition the Mp interface also carry the floor control requests and floor control responses from/to the floor control server.

The MRFP is connected to MRFC(s) through the Mp interface; it hosts the Mixer function and the floor control server function as defined in [25]. The Mixer is connected to the UE through the RTP/RTCP protocols and the floor control server is connected to the floor control client (hosted by the UE) through the Binary Floor Control Protocol as defined in [25]. The Mp interface is intended to carry the commands provided by the conference focus to the mixer and to send back events from the mixer, in addition the Mp interface also carry the floor control requests and floor control responses from/to the floor control server.

4.4.1 Functional split between the AS and MRFC

The following figure depicts the functional split between the AS and the MRFC; the MRFP is unchanged from previous section.

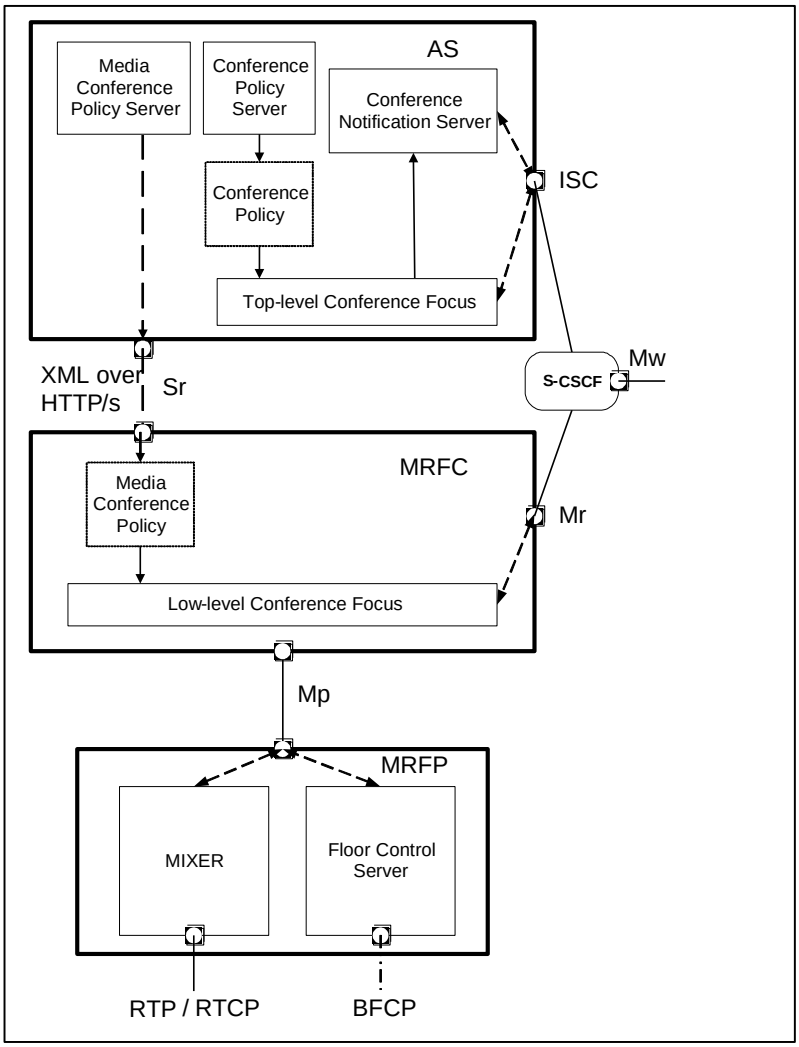


Figure 4.4.1.1: Functional split between the AS and MRFC

This model has been introduced in [19] in an IETF context, where the conference logic is split between two set servers:

AS

- It is seen as the top-level focus by the conference's participants, it is addressed by conference URIs.

- Implements the Conference Policy Server, thus acting as the logical function between the end-user and conference policies. This logical function is used by the end-user to subscribe to the conference service and also to modify its conference preferences.
- Execute the overall conference policies (Life-cycle, Membership, Authorization), except the media conference policy that is delegated to the MRFC.
- Might support a conference notification server using SIP SUBSCRIBE and NOTIFY mechanism as per [26].
- Might support advanced billing models: prepaid, postpaid, shared charging between participants, pay per conference, pay per codec, etc.

MRFC

- The low-level conference focus that is contacted only by top-level focus, this relationship is private.
- Load and execute the media conference policies (in addition to simple Life-cycle policy) that are dynamically fetched from the AS at the conference creation time.
- Control Audio/Video/Text mixers.
- Might generate Conference Detailed Records in XML format.

The focus as seen by the conference's participants defined in [19] is hosted by the AS, it is called the "top-level focus". The MRFC also hosts a focus logical function, but this focus is not directly addressed by the conference participants, only through the top-level focus. This "low-level focus" has limited actions:

- It cannot add a new conference participant or remove a participant on its own; this action is under the AS responsibility.
- Its main responsibility is upon reception of SIP INVITE to check that the Session Description Protocol offer or answer [27][28] matches with the media conference policy parameters (for instance the codec type or the codec bit rate). Based on that processing it can accept, reject or modify the participant's SIP session setup and control accordingly the mixer.
- It is important to notice that the low-level focus should be authorized to dynamically modify the multimedia session profile through SIP re-INVITE in order to fit with network condition changes (either reported by the AS or the UE, or reported by the MRFP).

The communication between the top-level and low-level focus(es) can use both delegation and protocol models, for example NETANN [3] for a simple conference and MSCML [15] for an advanced conference. NETANN or MSCML can be extended in order to carry the URL of the media conference policy by using the optional parameter of the SIP Request-URI, for instance:

```
sip:conf=1234@mrfc.hp.com;confpolicy=http://sipas.hp.com/policy/media-conf1234.xml
```

This allows the MRFC to dynamically fetch the media conference policy delegated from the AS at the conference creation time (and if necessary updates via mid-call XML).

This last section is aimed to provide a view on the advantages and drawbacks of the decomposed AS / MRFC model.

Advantages

- Better decoupling of role & responsibilities enabling fine grained scalability of either the AS or MRFC functions:
  - The AS is in charge of the conference application logic, in addition to notification service and conference policy server.
  - The AS does not have to deal with the underlying complexity of the SDP base format [27] plus the specific extensions for each audio/video codec, and the SDP Offer/Answer model [28].
  - One AS can use multiple MRFCs for complex or large conference scenario.
  - The AS can make a finer conference resource management, for instance by specializing MRFC for Audio or Audio/Video or Text.

- The MRFC is dedicated to load & execute the media conference policy and control the mixers accordingly.

- Better availability model, in case of MRFC failure, the AS can re-connect the participants to another MRFC instance.

- Allow the AS to focus on the overall service orchestration, like chaining of XDMS service and presence service with conferencing service.

- The AS can be located in a different network than MRFC(s).

Drawbacks

- Floor control message exchanges are not normalized between the MRFC and AS, The delay generated between the MRFC and AS nodes can degrade the user interactivity (one may argue here that some floor control delegation is also mandatory to preserve acceptable responsiveness to end-user's inputs).

4.5 AS programming and service implementation impacts on media server control

Two major models for AS programming and service implementation are:

- API model: the service is defined in terms of a programming language such as Java, C#, etc. Application frameworks such as J2EE or Parlay are frequently used to facilitate rapid development and integration with common components (e.g. with APIs towards databases and protocol stacks such as SIP and HTTP).

- Web model: the service is defined in terms of markup languages/scripts (e.g. VoiceXML, SCXML and CCXML). The scripts may be static - resident on the AS file system - or dynamic – in the latter case, techniques (such as JSP, ASP and Servlets) are used to dynamically generate script documents. Protocols such as HTTP are used to transmit the scripts.

Specific instance of AS services may incorporate elements from both models; for example, the Web model can use programming languages such as Java to implement some service logic in Servlets, while the API model can use Java to dynamically generate XML script documents for transport over SIP and/or HTTP.

While both the delegation model (Section 4.2) and the protocol model (Section 4.3) could be used with either AS programming model, the delegation models tends to be associated with the web model and the protocol model with the programming model.

The delegation model with its emphasis on 'coarse-grained' media commands fits well with an AS programming model where the media behaviour is defined in XML scripts (such as VoiceXML, SCXML and CCXML) which are independently executed on MRFC script engines. These scripts are located on the AS and retrieved by the MRFC using HTTP through the Sr interface. Scripts running on the MRFC can also provide notifications to the AS, as well as receive data updates through this interface. Furthermore, the AS can exercise more fine-grained control of the media behaviour by defining smaller scripts for the MRFC to execute so requiring the MRFC to retrieve further scripts from the AS through the Sr interface. In this way, the media flow and presentation state is delegated to the MRFC, while the AS retains overall control since scripts are defined (and can be dynamically generated) by the AS.

In the protocol model, the execution state of the service is implemented centrally by the program running in the AS and XML transmitted over the Cr interface provides fine-grained instructions to the MRFC – the AS retains more control over the media behaviour throughout service execution. However, the AS can choose to delegate some control to the MRFC by referencing or including scripts such as VoiceXML in its XML instructions.

Although there is a tendency to align the delegation model with the web model and the protocol model with the API model, the AS programming model does not significantly impact the choice of delegation versus protocol models, Both programming models can support use of HTTP to transmit resources and documents and both, to varying degrees, can support distributed control which in turn requires the MRFC to support script engines. If service developers prefer a single coherent environment for AS programming, then the delegation model with coarse-grained scripting (e.g. using VoiceXML, SCXML and CCXML) or the protocol model with relatively fine-grained XML messages (but no coarse-grained XML scripts, such as VoiceXML) are the most appropriate.

4.6 Packages, registration and extensibility

The capabilities of the MRFC need to be identified to the AS either directly as part of the control protocol itself or as a part of the AS-MRFC interaction model. Three issues relevant to capabilities are:

- Organization: Mechanisms by which MRFC functionality can be organized so that different MRFCs can support different media functionality. This mechanism could characterize functionality at two levels: major functionality (e.g. IVR only, conferencing only, IVR and conferencing) and the minor functionality (e.g. audio-video codec support, etc)
- Registration: Mechanisms by which an MRFC can advertise its supported (major and minor) functionality to interested parties, including ASs. This allows an AS to dynamically select an MRFC with the functionality appropriate to the service it wishes to run.
- Extensibility: Mechanisms by which an MRFC could define and advertise support for functionality which is non-standard or proprietary.

In the delegation model, functionality can be organized in terms of support for specific scripting languages. For example, an MRFC could specify that it support VoiceXML 2.1, SCXML 1.0 or CCXML 1.0 (where the standards themselves define which specific functionality must be supported by a conforming implementation). Later versions of these languages will provide more fine-grained organization using profiles defined in terms of functionality modules. If further fine-grained information needs to defined, a list of additional attributes (e.g. codec support, bargein support, transfer, etc) can be defined as part of a 3GPP profile for each language. None of these scripting languages provide builtin registration mechanisms. Therefore, an external mechanism needs to be identified for the delegation model to register MRFC capability, and support for this mechanism could be part of the 3GPP profile for these languages. Finally, extensibility is defined for script languages like VoiceXML using XML namespaces; i.e. the script can include properties, attributes and elements from other namespace which the script engine must ignore if it does not understand. Registration of extended capabilities would follow the same approach as for standard capabilities.

In the protocol model, some approaches use a mechanism whereby functionality is organized into packages (cf. H.248). For example, the SIP Control Framework [9] requires that functionality is organized into well-defined packages (such as ivr-basic, ivr-vxml, conference-basic, conference-advanced, etc). If more fine-grained capability information needs to be defined, a list of additional attributes (e.g. codec support, etc) can be defined as part of a 3GPP profile for each package. This protocol model approach provides a builtin mechanism by which a connecting AS can identify which functionality is supported. If capability registration is required prior to AS connection, then the approach advocates using standard SIP mechanisms (e.g. RFC 3840 "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)". Finally, this protocol model can support extensibility by defining new packages which incorporate the extended functionality.

4.7 MRFC acting as an RTSP client

AS may instruct an MRFC to control playing of a media resource located on an external server. Such external media resource may be specified by an RTSP URL. Such RTSP URL may be provided either over Sr (e.g. playing prompts specified in VoiceXML scripts), over Mr (e.g. playing announcements in RFC4240 [3]) or over Cr (e.g. using the SIP control framework [9][16]).

To access external media resources defined by RTSP URL MRFC performs the RTSP (RFC 2326 [30]) client functionality.

4.8 MRFC interacting with external media server

It may be beneficial for an MRFC to interact with a legacy media server to leverage their capabilities. Such media resource capabilities provided by external server may be accessible via SIP (e.g. playing announcements in RFC4240 [3]).

The profile for the SIP protocol used between MRFC and external media server is implementation dependent.

4.9 Recommendations

The above study gives the following conclusions for media server control:

- Both the delegation and protocol models described should be supported.

This study describes the two approaches and their differences in terms of command level and granularity, media execution state, network traffic, response times, support of cross network boundaries and programming models.

Both models are recommended to be supported as different media control mechanisms are appropriate for different service use cases and deployment architectures. Subclause 4.3.5 provides an example of the choice of model for different conferencing service use-cases.

- RFC 4240 [3] should be supported with the clarifications to VoiceXML invocation defined in the draft-ietf-mediactrl-vxml specification [4].

RFC 4240 [3] is a widely used, simple, quick, lightweight and easy way of invoking basic media processing capabilities as well as VoiceXML dialogs via SIP. There are some issues in the RFC 4240 [3] specification for VoiceXML invocation which make it problematic to build interoperable standards from. These issues are fixed by draft-ietf-mediactrl-vxml [4].

- VoiceXML should be used to specify all user dialogs.

VoiceXML is the most widely adopted and supported, international standard available for user dialogs. Subclauses 4.3.1, 4.3.1.4 and 6.2.1 describe how it can be used for media server control.

- A dedicated control channel should be supported.

This study describes the problems, in subclause 4.3.2 with using SIP and SIP INFO to carry media server control commands.

- Mid-call media processing should be supported (subclause 4.3.4).

- The AS/MRFC functional split as described in subclause 4.4.1 should be supported with a top-level focus, notification server and conference policy on the AS and with media conference policy handling and low-level conference focus on the MRFC.

Subclause 4.4.1 describes the advantages of this functional split which is compatible with the SIPPING and XCON conferencing models.

- The media server control methods should support organization, registration and extension of capabilities (subclause 4.6).

- RTSP URLs should be used (subclause 4.7) for services which require the sourcing of streamed media.

For the detailed specification of media server control for release 8 the following is recommended:

- The creation of a Cr reference point that can be used for both delegation and protocol models.

This reference point would support the functions of both the Cr and Sr interfaces (delegation and protocol models) described in this study. One reference point (with the combined functions) is proposed so that from an architectural view there is only one reference point between the AS and MRFC functional entities.

- The ways in which additional information for the advanced charging scenarios described in subclause 7.5.1 are passed across the Cr interface should be specified. The specification of the information itself however should be left for further study.

- The use of the VoiceXML 2.1 [6] standard or the VoiceXML 3.0 working draft is to be specified (the decision to be taken depending on availability of the VoiceXML 3.0 working draft).

- The use of RFC 4240 [3] and the draft-ietf-mediactrl-vxml [4] is to be specified.

- The specification of the AS/MRFC conferencing split and a corresponding media conference policy that can be used by a media server control method.

- The use of a SCXML [29] or CCXML [7] profile to use with the delegation model is to be specified.

- For the protocol model to align requirements with and specify the use of the protocol that is the result of the IETF mediactrl working group. If this work will not be available in the release 8 timeframe then specify the use of an existing protocol such as MSCML (RFC 4722) [15].

5 MRFC deployment scenarios

5.1 Introduction

The present section lists different MRFC deployment scenarios which, if required, may bring requirements for a media server control protocol.

5.2 Using different ASs to invoke, control and service the MRFC

In the simplest deployment scenario, the MRFC interacts with a single AS. However, an MRFC can also interact with multiple ASs where each AS carries out a different logical role in its interaction with the MRFC:

- Invoke role: AS invokes the MRFC by setting up a SIP signalling path between itself and the MRFC
- Control role: AS determines which service is to be executed on the MRFC
- Service role: AS provides service data and resources for the MRFC

These roles are correlated with the interfaces between the AS and MRFC. The AS which invokes the MRFC must support an ISC/Mr interface with the MRFC. The interfaces required by the control and service role depend on the choice of control protocol models.

In the delegation model where scripts are used to provide the service, the control AS uses a SIP interface to specify which script is to be executed; for example, VoiceXML, SCXML and CCXML scripts are referenced as part of the Request-URI in the invocating SIP INVITE; i.e. a single AS plays the invocation and control roles towards the MRFC. A separate AS can carry out the service role using the Sr interface where HTTP is used to transport scripts and other service data. This leads to a distributed AS architecture where there is a well-defined separation between a SIP AS which invokes the MRFC and controls which service to execute on it, and an HTTP AS which provides the MRFC with the content of the service itself. This architecture is shown in Fig 5.2.1.

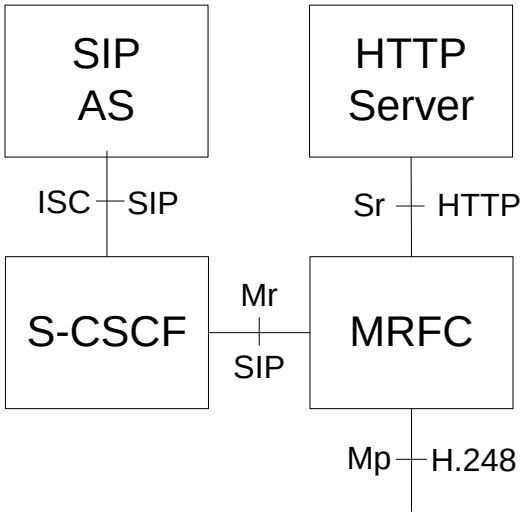


Figure 5.2.1: Multiple ASs with Delegation Model

There are some clear deployment benefits of this distributed AS architecture in terms of migration and cost. For operators with a service node architecture based on the web model, migrating to IMS only involves the addition of one new AS – a SIP AS which supports invocation and control roles. The service AS, on the other hand, can be exactly the

same as their Web Server in the service node architecture: i.e. an HTTP AS which supports resource (static or dynamic) retrieval via HTTP. Furthermore, the cost and complexity of HTTP ASs is typically far lower than a SIP AS.

With the protocol model using a dedicated control channel and a separate http server, each role can be carried out by a separate server. For example, with the SIP Control Framework and packages, one (SIP) AS invokes the service (ISC/Mr) and, using another SIP dialog, sets up the control channel, a second (Control) AS uses the dedicated control channel (Cr) to pass control package messages, and a third (HTTP) server provides resources (e.g. audio files, VoiceXML scripts) over HTTP (Sr). This illustrated in Figure 5.2.2.

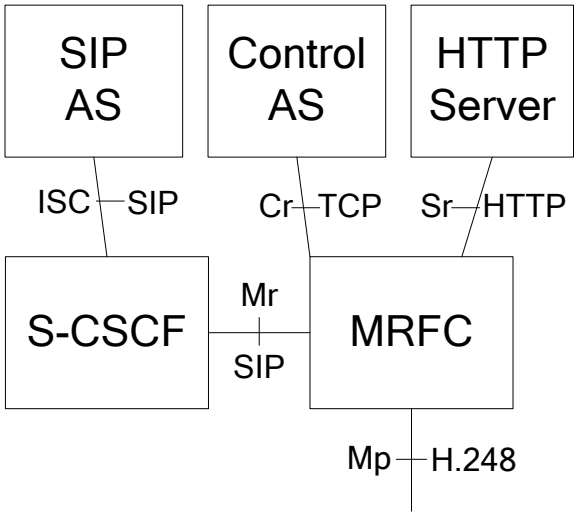


Figure 5.2.2: Multiple ASs with Protocol Model and a separate HTTP server

Note that this functional model could, of course, be simplified in actual deployment, so that the roles are collapsed into a single AS or two ASs (e.g. one SIP AS for invocation, another Control/HTTP AS for control and service; or one SIP/Control AS for invocation and control, and other HTTP AS for service).

If the control and service role are collapsed into a single AS then the Cr interface can be used as illustrated in Figure 5.2.3.

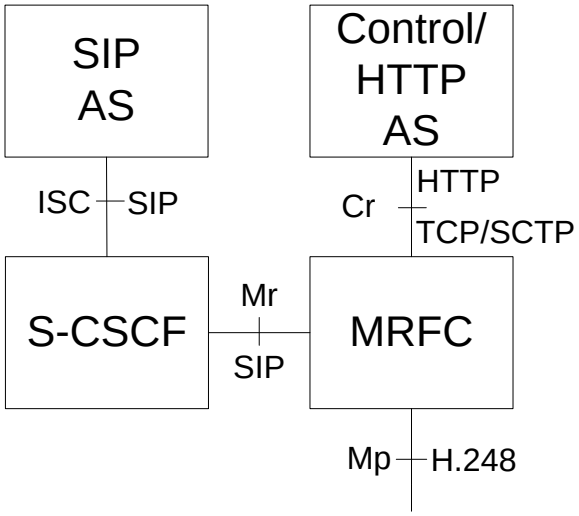


Figure 5.2.3: Multiple ASs with protocol model – one AS performing control and service roles

5.3 AS in one network controlling an MRFC in a different network

The interfaces between the S-CSCF, AS and MRFC are interfaces that may cross network boundaries.

When IMS users are roaming in a visited network their services and service logic are controlled from their home network.

When there is a need for a service to engage media resources, two main scenarios can be envisaged - the resources might be engaged from, or close to, the home network (home MRFC), or might be engaged from, or close to, the visited network (visited MRFC). There are currently no specifications indicating which model is to be used.

Engaging the resources at the home network is the simplest model but the drawback is that media always needs to be carried between visited and home networks; this effort may be not justified when the needed media resources are basic (e.g. user interactions with announcement and DTMF) or when there is a significant cost saving (e.g. conference with all the parties in the visited network).

Engaging the resources at the visited network optimizes the media interactions between networks and simplifies end to end QoS network accountability (only the local operator is involved).

When the AS is in a different network from the MRFC the delegation model is beneficial with respect to the protocol model as it enables a reduction of interactions across network boundaries (for example between the home AS and the visited MRFC).

Having ASs, S-CSCFs and MRFCs in different networks has a number of other implications and issues:

- The MRFC will have to send charging records and authorizations to several networks, for example the home network (for services charging) and to the visited network (for network usage charging). This implies that the MRFC Ro and Rf charging interfaces must be able to cross network boundaries.
- Appropriate security mechanisms must be in place as the MRFC, S-CSCF and AS are not in the same trust domain (for example to pass voucher or PIN numbers).
- How does the AS discover or find the MRFCs in another network?
- How does the AS obtain the information to select the most appropriate MRFC to use?

The above scenarios are not incompatible and might coexist depending of the established agreements between network providers and depending of the service relevance.

5.4 Several ASs controlling one MRFC, one AS controlling several MRFCs

It is not explicitly specified, although widely assumed, that an MRFC can be controlled via several AS's. This should be explicitly stated when specifying the relevant interfaces between these core network elements.

This has the following consequences on the media control interfaces:

- Resource isolation – one AS's use of an MRFC could not deny required MRFC resources from another AS.
- Privacy – one AS could not be able to obtain unauthorized information about another AS's use of an MRFC.

Similarly it is not explicitly specified, although widely assumed that one AS can control several MRFCs. This should be explicitly stated when specifying the relevant interfaces between these core network elements.

The AS may support a mechanism to request and combine capabilities provided by multiple MRFCs to fulfil media control functions requested by the application in case there is no single MRFC that can fulfil all the requested media control functions.

This has the following consequences on the media control interfaces:

- Discovery – the AS could have a mechanism available to discover the MRFCs and their media capabilities. The media control interfaces may be involved in this mechanism (or separate independent mechanisms used).
- Routing – the AS could have a mechanism available to select the most appropriate media server to use for a service. This selection process could be based on information like the most efficient network usage or the core network or network element load. The media control interfaces may be involved in this mechanism (or separate independent mechanisms used).

5.5 Core Network elements other than the AS invoking MRFC media processing capabilities

It may be required for core network elements other than an AS to invoke MRFC capabilities if defined as part of their specifications.

A common use case of a core network equipment making use of MRF processing equipment is for announcements or transcoding due to error cases or incompatible terminal equipment.

This use case is best supported by:

- A simple way to invoke an MRFC and a set of basic capabilities to be supported.

5.6 Intermediary broker function between AS and MRFC

5.6.1 General

This section is aimed to introduce architectures about the intermediary broker function with MRB. The IETF draft [34] describes the MRB which implements the function of the MRFC selection. There are two models of MRB, 'in-line' and 'query' MRB.

As defined in the draft-boulton-mediactrl-mrb [34], the Media Resource Broker (MRB) is a functional entity that is responsible for both collection of appropriate published MRF information and supplying of appropriate MRF information to consuming entities such as the AS.

The benefits to introduce MRB in to the network architecture are:

- Simplify the configuration and operation of the AS.
- Support dynamic upgrade of MRFs without impact on and synchronization with the AS.
- Simplify the deployment of MRFs by aggregation of the MRF status and capability to the MRB.
- Allow more efficient sharing of the resources of the MRFs by multiple AS.

NOTE: The MRB can be co-located with other entities, e.g. AS or S-CSCF or other entities, or stand alone.

5.6.2 Architecture Requirements

The following are the requirements for an appropriate Media Resource Broker architecture.

a) The MRB shall support the sharing of a pool of heterogeneous MRF resources by multiple heterogeneous applications.

b) It shall be possible for an application to specify to the MRB what MRF attributes and corresponding values it requires for supporting a call/call leg. The attribute list is not defined further in the present document.

c) It shall be possible for the MRB to allocate MRF resources across different applications according to some kind of rules/policy. Consequently, an application shall indicate some sort of application identifier in its request to the MRB (so that the MRB can allocate a fair share of resources per application).

d) It shall be possible for the MRB to use per-application and per-subscriber SLA or QoS type of information in its resource selection process.

e) It shall be possible for the MRB to calculate the remaining capacity of a particular MRF resource based on the nature of current usage. This implies that MRB may have a capacity model for a particular kind of MRF resource, which, for example, may be able to handle 20 IVR calls at once with only DTMF input, or 5 at once with spoken word input, or some mix.

f) It shall be possible for an application to send (i.e. SIP INVITE) the call/call leg to the MRF resources identified for handling it.

g) It shall be possible for an application to request resources for multiple calls/call legs with a single request; for example, for conference legs for a conference it may be necessary, or for a Freephone application it may be desirable for efficiency purposes.

h) It shall be possible for an application to decrease the amount of resources previously assigned to it (especially useful for conferencing; also for Freephone scenarios that don't surge as much as expected). Likewise, it shall be possible for an application to increase the amount of resources it had previously been assigned (especially for conferencing, but also Freephone scenarios that surge more than anticipated).

i) It should be possible for an application server to indicate alternative & preferential MRF resource attribute/value sets (if an application can't have a MRF resource with attribute set {A} then it would prefer set {B} else {C}). In this case where AS expresses alternative possibilities of attributes for MRF resources, it shall be possible for the AS to be informed of what is being provided, so the AS can control the MRF resource appropriately.

j) It shall be possible for an application to be the entity that determines when a media resource is no longer needed for a call/call leg (for example, a conference caller may drop off because of a meeting break, or to briefly conduct some other business, but the port should be left devoted to that conference).

k) The MRB functionality shall not be involved in understanding or participating in any way in AS-MRF media control protocols (efficiency, separation of concerns).

l) It shall be possible for MRB to know the following:

- Available MRF resources and their attributes; current and future. This may take into account planned and unplanned downtime and the scheduled addition or availability of more MRF resources.
- Rules for fair-share allocation across applications.
- Per-application and per-subscriber SLA and QoS criteria.
- Capacity models for particular MRF resources.
- Reservations for future use of media resources (for example, for conferencing, or for anticipated traffic spikes for other applications such as a FreePhone application).

m) The MRB shall be able to re-synch on an MRF's resource status.

5.6.3 Architecture Alternatives

There are four possible application architectures of MRB in IMS: In-Line Model –one, In-Line Model –two, In-Line Model – three (AS directly to MRB), Query Model.

1. In-Line Model –one

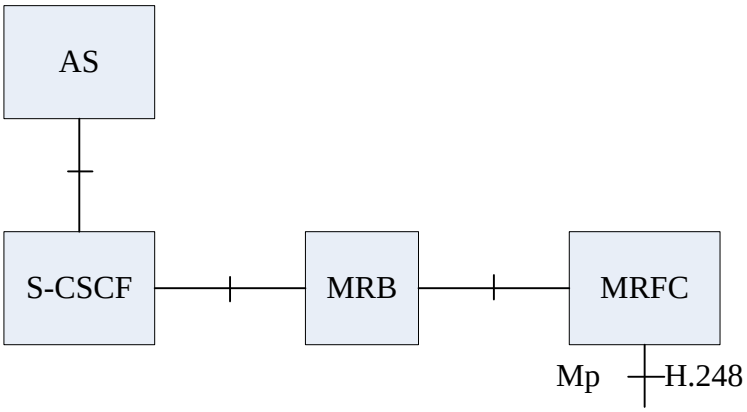


Figure 5.6.1: In-Line Model –one

In this model, the MRB is between the S-CSCF and the MRFC. The AS sends message passing the S-CSCF to the MRB. The MRB selects an appropriate MRFC according to the information in the message. The MRFC publishes status information and makes register and deregister procedure.

2. In-Line Model –two

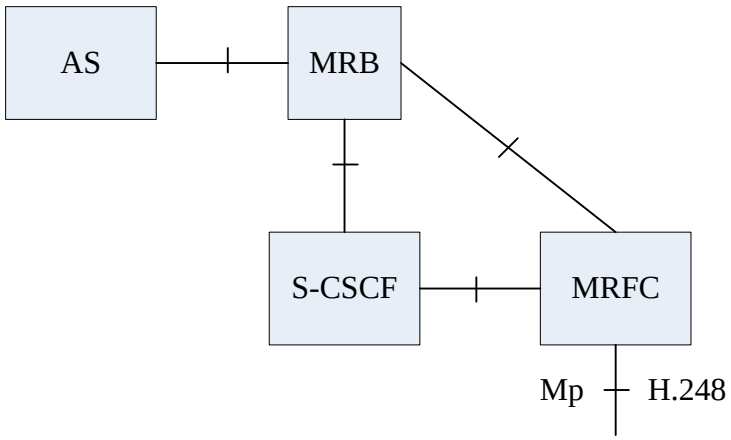


Figure 5.6.2: In-Line Model –two

In this model, the MRB is between the AS and the S-CSCF. The AS sends message to the MRB, MRB selects an appropriate MRFC according to the information in the message and sends message to the S-CSCF, the S-CSCF routes the message to the MRFC.

3. In-Line Model – Three (AS directly to MRB)

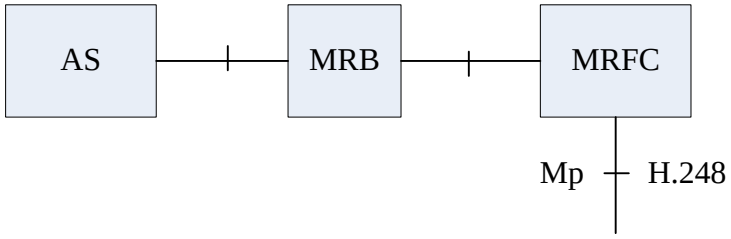


Figure 5.6.3: In-Line Model – three (AS directly to MRB)

In this model, the AS directly connects to the MRB, the AS sends message passing the MRB to the MRFC.

4. Query Model

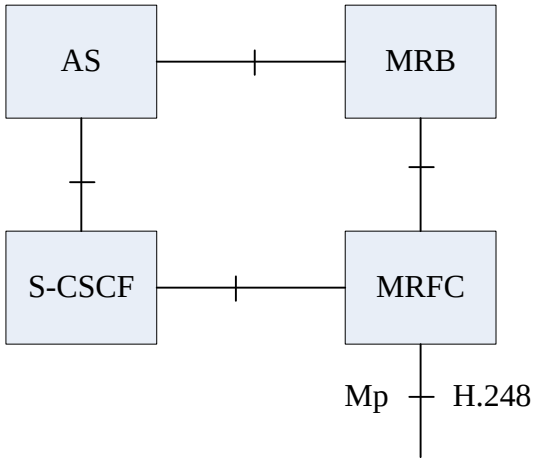


Figure 5.6.4: Query Model

In this model, the AS queries the MRB to get the information of the MRFC status and capability to make a decision which MRFC will be selected. If the AS selects a MRFC, the AS sends message passing the S-CSCF to the MRFC.

The introduction of MRB has the following consequences on the media control interfaces:

- Media resource capability publication – the MRF could publish its status and capability to the MRB.
- Media resource capability query – the MRB could query the status and capability of MRF.

5.6.4 Architecture analysis

There is nothing to preclude an MRB from being a distributed or hierarchical system, however that level of granularity is left for future study.

5.6.4.1 AS resource requests to MRB

The Query model can meet all the requirements listed. The In-Line model is less versatile (it doesn't support fully requirements g, h, i, and j in subclause 5.6.2), but is simpler (it doesn't require a new interface, just payload in a SIP message), and could suffice for most IVR and simple conferencing applications.

The Query model more straightforwardly allows an application/AS to know the address of the assigned MRF resources for the purpose of using a separate control channel. The Query model also allows a conferencing AS to more straightforwardly setup a single large conference that is spread and linked across multiple MRF physical resource units.

Regarding the In-Line sub-options presented in subclause 5.6.3 above, the "In-Line Model - two" has an advantage over the other sub-options ("-one" or "-three") because it allows the use of native routing capabilities of the S-CSCF to route a call (i.e. its SIP signalling) to a MRF resource just as it would any other endpoint. Note that MRF resources could be inside the same network as the S-CSCF or external to it. A slight advantage of "-one" over "-two" is that in "one" an AS wouldn't have to be aware of the MRB as a separate entity. Overall, "-two" seems the better arrangement.

It is important to note a synergy between the Query and the In-Line models: the same attribute description/format could be used by an application/AS in either case. It is simply payload in a message.

5.6.4.2 MRB knowledge of MRF resource-related information

The same solution can be used for both the Query and the In-Line models to procure knowledge of MRF resource-related information. This is another key way in which the models are synergistic.

Some information would most naturally come from the operations environment, so minimally an operations interface to an MRB is needed, for example for MRF resource downtime, application fair-share rules and conference reservation requests. That same operation interface could be used to convey to MRB knowledge of MRB resource characteristics.

Alternatively, there could be some kind of MRB-MRF resource interface through which MRB somehow discovers MRF resources and their characteristics.

Having all of the information come through the same operations interface may be a simpler starting point.

5.6.4.3 Other Application Considerations

In the case of a large conference that requires the use of multiple physical MRF resources, it would be the responsibility of the conferencing AS to link and control those MRF resources.

5.6.5 Conclusion

Pursue both the Query model and the In-Line model. The Query model may be necessary from some service/network providers, but overkill for others. The In-Line model may suffice for some service/network providers.

Among the In-Line model sub-options, "In-Line model-two" that is sequenced AS - MRB - S-CSCF - MRF is most useful.

Synergy between the two approaches should be exploited, as mentioned above (MRF resource attribute characterization, and MRB knowledge of MRF resources).

Initially, the use of an operations interface for MRB knowledge of MS resources and reservations is the simplest starting point. A direct MRB-MRF interface may have merits especially for dynamic information exchange, and warrants further consideration in a future release.

The MRB is an optional functional element.

NOTE: Simpler options exist that could in fact be used, meeting even fewer requirements or architecturally awkward (for example, embedded in the routing function of the S-CSCF), so such things have not been discussed in this section.

5.7 Functionality of media resource composition

Advanced multimedia services may require complex media resource capabilities. For example, an advanced multimedia conference service may require mixing, recording, playback, TTS and etc. A single MRF may not always be able to support all the capabilities required by a complex application, also it may be beneficial to be able to leverage the capabilities provided by MRFs already deployed instead of upgrading exiting MRFs or deploying new MRFs. In this case, composition of media resource capabilities provided by multiple MRFs is needed.

The functionality of media resource composition is to combine media resource capabilities provided by more than one MRF to fulfil media control request from the application.

The media resources that may be composed are subject to the applications perfomed by the AS and capabilities of MRFs. One basic principle is that there shall be no difference for the user experience when media resource composition is performed in the network.

The location to implement the functionality of media resource composition can be flexible in deployment. Several possible locations to implement the functionality of media resource composition are described below:

1: media resource composition in the AS

In this case, there is no single MRF that can provide all the required media resource capabilities, the AS can implement media resource composition by controlling multiple MRFs and combining their media resource capabilities to fulfil the request from the application.

2: media resource composition in the MRB

In this case, it is assumed that MRB is deployed in the network and the AS requests media resource control function through an MRB. In case there is no single MRF that can provide all the required media resource capabilities, the media resource composition functionality can be implemented in the MRB and it will interact with multiple MRFs and combine their media resource capabilities to fulfil the request from the AS.

Wherever the location is, the media resource composition functionality needs to interact with the AS and other MRFs to fulfil its composing requirement.

When interacting with other MRFs, the interfaces between the media resource composition functionality and MRF can be equivalent to Mr and Cr.

When interacting with AS or other entities such as S-CSCF, the interfaces between AS and media resource composition functionality can be equivalent to Mr, Cr and Sr.

When co-located with other entities, the interface between media resource composition functionality and the co-located entity is an internal interface.

5.8 Recommendations

The above study gives the following conclusions for media server control:

- From subclause 5,2, the separation of the invoking, controlling and service data media control interfaces should be possible.
- From subclause 5.3, when crossing network boundaries the delegation model has advantages with respect to the protocol model.
- From subclause 5.4, the media server control must have mechanisms for resource isolation and privacy between requests from different applications and application servers.

- From subclause 5.6, the media resource broker function should be introduced into the 3GPP architecture and procedures, allowing for both the Query and In-Line models (the 3GPP SA2 group would have to be consulted for the introduction of the MRB functional entity and associated reference points). For the present, it should be assumed that MRB knowledge of MRF resources occurs through an operations type interface. Architecturally, this is recommended for release 8. The detailed interface specification work for the MRB could be looked for from the IEFT mediactrl working group whose charter scope allows for such work. The MRB requirements in subclause 5.6.2 should be included in the IETF work. The actual interface specifications are unlikely to be available in the release 8 timeframe. A direct MRB-MRF interface whereby MRB can be aware of existing MRF resources and status should be considered for a future release, where again the IETF mediactrl working group could be looked to for an actual interface specification.

- From subclause 5.7, the media resource composition functionality is a logical function which can be co-located with various entities in the 3GPP IM CN Subsystem. There is no protocol requirement identified in the study.

6 Relevant Specifications

6.1 Introduction

The present section lists existing standards, RFC's or published specifications relevant to the study of media server control protocols together with a brief description of the work and its relevance.

6.2 Standards and draft standards

6.2.1 VoiceXML

VoiceXML [5][6] is an XML scripting language for interactive media functionality.

The language defines an extensive set of tags which cover output functionality (media files and speech synthesis), input (DTMF, speech recognition and recording), logic (if-then-else), data model (scoped variables), events (noinput, nomatch) as well as a well-defined dialog algorithm (FIA) which manages a flow of input-output transactions. The language allows external resources – for example, DTMF or speech recognition grammars – to be specified in the VoiceXML document and fetched using the Sr or Cr interface. Depending on the flow of the interaction, further VoiceXML documents can be fetched and control transferred to the fetched document. VoiceXML also allows data to be passed to the application plane entity when a VoiceXML document or resource is fetched. VoiceXML supports both simple and complex interactive media behavior.

The current version, VoiceXML 2.0 [5], is W3C Recommendation (standard) which has extensive industry support and existing commercial deployments in the telecom sector. It is also supported by most IETF informational and working draft proposals (RFC4240, draft-ietf-mediactrl-vxml, MSML, MSCP, SIP Control Framework) for media interaction. W3C is also actively developing this standard with VoiceXML 2.1 [6] due out soon and VoiceXML 3.0 on the horizon.

VoiceXML does have some issues which may need to be addressed in the MRFC context. Firstly, if interactive video capability is an MRF requirement, then VoiceXML 2.0 has no explicit support. However, as described in http://www.voicexmlreview.org/Mar2006/features/video_interactive_services.html, this can be largely addressed in the current version without compromising interoperability and VoiceXML 3.0 is expected to explicitly addressing it. Secondly, VoiceXML has tags which allow the caller to be transferred (blind or bridged) to another telephone destination. This may be problematic if an MRF is not permitted to generate outbound calls. However, this feature of VoiceXML is optional and could be addressed by a VoiceXML profile for the MRFC use case. To overcome this limitation the MRFC may request the AS to initiate outbound calls or call transfers on its behalf. Finally, there may be cases where 3GPP wishes to extend VoiceXML with additional or different functionality. W3C have recognized this type of VoiceXML usage and VoiceXML 3.0 is expected to have a modularization framework which allows profiles, including a media server profile, and new languages to be defined.

In summary, the key benefits of VoiceXML are that it is an existing, well-supported, international standard and provides the interactive media functionality required in an MRFC context.

6.2.2 CCXML

CCXML [7] is a W3C XML scripting language for conferencing and call control functionality which was designed to complement VoiceXML's interactive media functionality. The language uses an event-driven algorithm where user-defined actions are triggered when events are fired.

The CCXML language provides tags for 4 areas of functionalities, Firstly, it can receive inbound calls and create outbound calls using a model which supports telephony definitions, such as JAIN Call Control, and which supports various telephony protocols including SIP. When an incoming call is received, an alerting event is generated and the script can specify actions to perform, including extracting information from the call signaling, accepting or rejecting the call. CCXML also has a tag to generate an outbound call where the script can specify the telephony protocol, destination URI, a-number, etc. The second area of functionality is dialog management: CCXML has tags to prepare, start and stop dialogs. For example, when the incoming call is in an alerting state, the script could specify that an 'early media' VoiceXML dialog is to be started. The various states of the dialog are indicated by events. Thirdly, CCXML supports conferencing functionality: there are tags for creation and destruction of conferences, as well as tags for adding and removing participant SIP connections to/from the conference. Finally, CCXML has Input Output functionality which allows it to send and receive events to/from internal sources (connections, dialog and conferences) and external sources (this is in additional to functionality which allows fetch and transition to CCXML documents just like VoiceXML). One such functionality allows CCXML scripts to send data to and receive data from HTTP servers.

CCXML fits well with the delegation model. The CCXML script to execute is specified in the SIP INVITE received on the Mr interface; for example,

```
INVITE sip:control@mrf.example.com;ccxml=http://server.example.com/conference.ccxml    SIP/2.0
```

The CCXML script would then be fetched with HTTP using the Sr interface. Upon execution of the script, the CCXML fires an event indicating that an incoming call (the SIP connection) is an alerting state and the script can then specify that a multi-party conference is to be created, an announcement played to the UE (using VoiceXML), then the UE is joined to the conference; for example,

```
<ccxml version="1.0" xmlns="http://www.w3.org/2002/09/ccxml">
<var name="connection" expr="''"/>
<eventprocessor>
     <transition event="connection.alerting" name="evt">
        <assign name="connection" expr="evt.connectionid"/>
         <createconference id="conf1" />
    </transition>
    <transition event="conference.created">
         <accept/>
    </transition>
   <transition event="connection.connected">
       <dialogstart src="'http://vxmlserver.example.net/welcome.vxml'"/>
   </transition>
    <transition event="dialog.exit" name="evt">
        <join id1="connection" id2="conf1"/>
   </transition>
</eventprocessor>
</ccxml>
```

For each UE to be added to the conference, the AS/CSCF would reference the same CCXML script in the SIP INVITE sent to the MRFC. In that way, each participant would hear the same announcement – specified in the welcome.vxml script - and then joined to the same conference – conf1. The script can be easily extended so that script interacts with a conference focus over the Sr interface (e.g. to obtain conference policy information, indicate active talkers, etc).

The current version, CCXML 1.0, is W3C Last Call Working Draft (i.e. it is on W3C Standards track but not yet attained Recommendation (standard) status). It is expected that CCXML will become a W3C Recommendation by 2007. As an emerging W3C specified, CCXML has limited but growing industry support; support is strongest with companies which also use VoiceXML.

CCXML does raise a number of issues which need to be addressed for its use as a MRFC script language. Firstly, if video conference is an MRF requirement, then CCXML 1.0 has no explicit support. However, this can be largely addressed in the current version without compromising interoperability; for example, defining a 3GPP profile where conference policy information is retrieved using HTTP from a conference focus, and the information is used to create the conference and its video layout. More explicit control over media streams, including where to place the UE's video in the layout, can be addressed by the addition of stream control tags (analogous to the <stream> element in MSML and MSCP). Secondly, CCXML has a tag which allows outbound calls to be created and then joined to the conference. This may be problematic if an MRF is not permitted to generate outbound calls. If this feature is not allowed in an MRF,

then the 3GPP profile could explicit disallow it. To overcome this limitation the MRFC may request the AS to initiate outbound calls on its behalf. Thirdly, there is currently no specification which describes how CCXML scripts are specified in SIP INVITEs and managed over the Mr interface. This could be remedied with a simple specification which provides for CCXML what draft-ietf-mediactrl-vxml provided for VoiceXML.

Finally, CCXML permits telephony protocols other than SIP. Inbound and outbound ISUP calls could be received by CCXML, depending on the implementation. CCXML is relatively agnostic on this issue: it doesn't specify which protocols are essential for its implementation. Consequently, a 3GPP profile for CCXML could specify that only the SIP protocol is to be supported.

In summary, the key benefits of CCXML is it is an emerging script standard which fits the delegation model and provides the call and conferencing functionality required in an MRFC context.

6.2.3 SCXML

SCXML [29] is a general-purpose event-based state language based on Harel State Tables and incorporating some CCXML concepts.

Harel State Tables (as used in UML) provide a mathematically sound semantics for state machine notations. SCXML provides an XML representation of state machines and adheres to the Harel State Table semantics, thereby providing powerful and compact control abstracts. Capabilities include representation and processing of state machines with support for composite states, parallel states, history states, etc. SCXML inherits from CCXML a data model, asynchronous data submission, executable content and service invocation.

Unlike CCXML, SCXML provide no built-in functionality beyond the ability to invoke (external) components, as well as send event messages to, and receive messages from, such components. SCXML is being used in media applications to control and coordinate VoiceXML and multimodal components, as well as non-media applications to coordinate between web services or components both for network services and clients (for example, SCXML is part of Apache Shale, a web application framework based on JavaServer Faces). SCXML is also extensible: new functionality can be added using a profile which defines new elements; for example, it could be extended with (CCXML) tags to provide call handling and conferencing support.

For the delegation model, SCXML is an alternative to CCXML as a MRFC script language to manage and choreograph capabilities running on the MRFC; for example, to start an IVR dialog and, when it is complete, add the participant to a conference. SCXML has the benefit that no media functionality is built into the language (unlike CCXML), so an MRFC profile can be built which would fit the requirements without bringing unwanted features. For example, a MRFC profile might include media support for inbound calls, IVR dialog invocation and multimedia conferencing only. Furthermore, the profile can specify whether these capabilities are part of the language (e.g. adding a <conference> tag) or required components which a SCXML script can <invoke> and send/receive messages.

As an example of how SCXML could coordinate VoiceXML and multimedia conferencing, assume a MRFC profile where SCXML support additional tags for <accept> ing and <disconnect>ing incoming calls and call alerting events (cc namespace) as well as tags and events for creating and manipulating multimedia conferences (mc namespace). The SCXML script to execute is specified in the SIP INVITE received on the Mr interface; for example,

```
INVITE sip:control@mrf.example.com;scxml=http://server.example.com/conference.scxml    SIP/2.0
```

The SCXML script would then be fetched with HTTP using the Sr interface. Upon execution of the script, the SCXML fires a cc 'alerting' event indicating that an incoming SIP call is an alerting state and the script can then specify that a multi-party conference is to be created, an announcement played to the UE (using VoiceXML), then the UE is joined to the conference; for example,

```
<scxml xmlns="http://www.w3.org/2005/07/scxml" xmlns:cc="http://www.example.com/mrfc-
profile/10/callcontrol" xmlns:mc="http://www.example.com/mrfc-profile/10/conferencecontrol"
version="1.0" initialstate="idle">
<datamodel>
<data name="connection" expr="''"/>
</datamodel>

<state id="idle">
  <transition event="cc:connection.alerting" target="CreateConference">
    <assign location="/data[@name='connection']"
            expr="_eventdata.connectionid"/>
  </transition>
</state>
```

```
<state id="CreateConference">
  <onentry> <mc:createconference id="conf1" /> </onentry>
  <transition event="cc:conference.created" target="callConnecting">
    <cc:accept/>
  </transition>
</state>

<state id="callConnecting">
   <transition event="connection.connected" target="playDialog"/>
</state>

<state id="playDialog">
  <invoke src="'http://vxmlserver.example.net/welcome.vxml'"
          targettype="application/voicexml+xml"/>
  <transition event="dialog.exit" target="inconference">
    <mc:join id1="connection" id2="conf1"/>
  </transition>
</state>
</scxml>
```

6.2.4 Explanation of Mp interface

6.2.4.1 Introduction

Valuable deployment experience can be gained by studying the Mp interface and the usage of H.248 protocol as a basis for the protocol model. This section describes the H.248 protocol.

H.248 (also known as the Megaco protocol) is the standard for allowing a Multimedia Resource Function Control (MRFC) to control Multimedia Resource Function Processor (MRFP). The megaco protocol is a result of joint efforts of the IETF and the ITU-T Study Group 16. The protocol definition of this protocol is common text with ITU-T Recommendation H.248.

H.248 protocol was designed with separation of control plane and the media plane So anything that needs media processing can be controlled using H.248, control of conference equipment, announcements, TTS, ASR, etc.

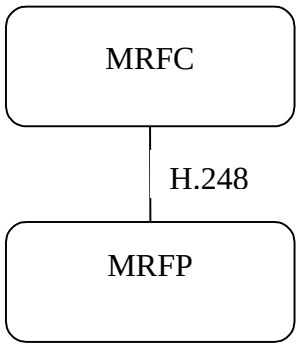


Figure 6.2.4.1.1: H.248

6.2.4.2 Main Characteristics

Protocol format is flexible with two formats available binary ASN.1 based and text ABNF based. Following trends from other text based protocols, namely SIP, H.248 has been used more text based. The advantage with text based format is that it is easier to read making it easy to trouble shoot and develop against.

The protocol is very efficient and does not require extra overload or heavy syntax.

The protocol is transport independent both from transport and media processing. H.248 has been used in ATM and IP networks. The IP network may be IPv4 and v6 based. The media types that can be controlled with H.248 is far ranging, RTP, UDP, TCP, MSRP, etc. This is a key feature which makes H.248 flexible with any future development.

The protocol syntax has several benefits making it a robust protocol, these are the key ones:

- Simple commands for processing requests (Add, Modify and Subtract commands)

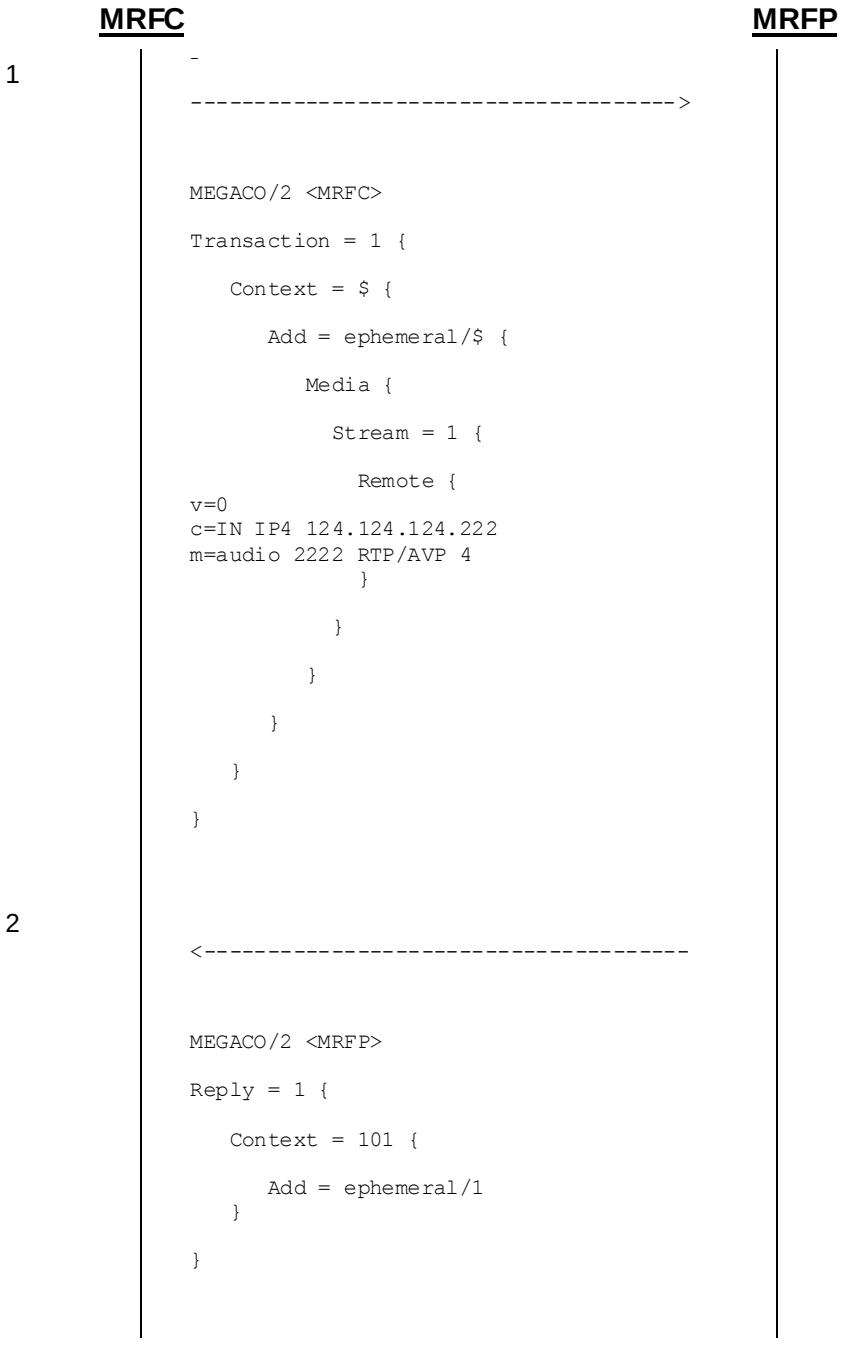
- Support of virtual MRFP's allowing multiple MRFC's controlling same equipment.
- Under disturbance scenarios it is possible to verify that entities are aligned with Audit commands

6.2.4.3 Example

There are numerous packages standardized by ITU-T, 3GPP and TISPAN that were created to address different requirements. New packages continue to be defined.

A package may cover the following area: properties, signals, events, observed events and statistics. These cover all areas of media processing.

The following sequence shows how H.248 provides announcement support using packages defined in H.248.9 standard, ie, "an" package. The highlighted line in step 3 indicates how announcement is applied while the rest is part of H.248 core protocol.



3

```
------------------------------------>


MEGACO/2 <MRFC>

Transaction = 2 {

   Context = 101 {

      Mod = ephemeral/1 {

         Media {

           Stream = 1 {

             LocalControl {

               Mode = SendOnly
               an/apf(an=x,di=ext)

             }

           }

         }

      }

   }

}
```

4

```
<------------------------------------


MEGACO/2 <MRFP>

Reply = 2 {

   Context = 101 {

      Mod = ephemeral/1

   }

}
```

6.3 RFC's

6.4 Informational RFC's

6.4.1 RFC 4240 ('netann')

RFC 4240 [3] (also known as 'netann') provides a mechanism for invocation of basic media services on the MRFC using SIP. RFC 4240 defines three services:

1. Announcement: play a media resource to the SIP connection
2. Dialog: invoke a VoiceXML dialog to the SIP connection
3. Conferencing: join the SIP connection to a simple conference

A service is invoked by means of the SIP INVITE Request-URI: the user part indicates the service, and additional URI parameters can be specified to configure the service. If the MRFC supports the service, and the service parameters are acceptable, the service is initiated when the ACK message is received and continues until a BYE message is sent or received. If the MRFC does not support the service, or there is a problem with the parameters or resource, the MRFC returns the appropriate SIP error status codes.

The announcement service is invoked by a Request-URI with user portion "annc" and URI parameters controlling the content and delivery of the announcement, including the mandatory parameter "play" indicating the resource to play. For example:

sip: annc@mrfc.example.com;play=http://asexample.com/welcome.wav

If the resource "http://as.example.com/welcome.wav" cannot be found or retrieved, error codes are returned. Otherwise, the resource is played to completion and then a BYE is returned.

The dialog service is invoked with a Request-URI with user portion "dialog" and URI parameters controlling the content and delivery of the dialog, including the mandatory parameter "voicexml" indicating the VoiceXML script to execute. For example:

sip: dialog@mrfc.example.com;play=http://vxml.example.com/promptandcollect.vxml

Again, error codes are returned if the script cannot be found. Otherwise, the VoiceXML script at http://vxml.example.com/promptandcollect.vxml is executed.

The conference service is invoked with a Request-URI with the user part "conf-<uniqueid>" where ‘uniqueid" identifies a unique conference mixing session. For example:

sip: conf-id100@mrfc.example.com

If the conference session identified by the URI "conf-id100@mrfc.example.com" does not exist on the media server but conferencing resources are available, then the MRFC creates a new mixing session and the SIP UE is joined to the conference. If a conference already exists, then the UE is joined to the conference mix. A UE leaves the conference by issuing a BYE on its SIP dialog. A conference session exists as long as there is at least one SIP dialog joined to the conference.

RFC4240 is well accepted in the industry due to the simplicity of how these media services are identified and invoked.

This simplicity, however, requires RFC4240 to be augmented with other specifications to attain desired MRFC functionality, especially for conferencing and IVR. Media languages like MSCML, MSML and MSCP go beyond RFC4240 in order to provide advanced conferencing services which support explicit conference setup, etc. Moreover, its description of the VoiceXML dialog service is incomplete: the current description is insufficient for interoperable implementations. In particular, the VoiceXML dialog service needs to address issues such as how Request-URI parameter data is mapped into VoiceXML, how data can be sent to the AS mid-call, how media support is achieved and how VoiceXML’s (optional) outbound calling functionality is addressed. The draft-ietf-mediactrl-vxml specification [4] builds on RFC4240 to address these issues (see subclause 6.5.1).

6.4.2 RFC 4722 (‘MSCML’)

Media Server Control Markup Language (MSCML) is a markup language used in conjunction with SIP to provide advanced conferencing and interactive voice response (IVR) functions. MSCML presents an application-level control model, as contrasted to device-level control models.

The RFC describes the Media Server Control Markup Language (MSCML) and its usage. It describes payloads that one can send to a media server using standard SIP INVITE and INFO methods and the capabilities these payloads implement. It builds on the RFC 4240 [3] use of media server SIP URI formats.

Prior to MSCML, there was not a standard way to deliver SIP-based enhanced conferencing. Basic SIP constructs, such as those described in RFC 4240 [3], serve simple n-way conferencing well. The SIP URI provides a natural mechanism for identifying a specific SIP conference, while INVITE and BYE methods elegantly implement conference join and leave semantics. However, enhanced conferencing applications also require features such as sizing and resizing, in-conference IVR operations (e.g., recording and playing participant names to the full conference), and conference event reporting. MSCML payloads within standard SIP methods realize these features.

The structure and approach of MSCML satisfy the requirements set out in RFC 4353 [19]. In particular, MSCML serves as the interface between the conference server or focus and a centralized conference mixer. In this case, a media server has the role of the conference mixer.

There are two broad classes of MSCML functionality. The first class includes primitives for advanced conferencing, such as conference configuration, participant leg manipulation, and conference event reporting. The second class comprises primitives for interactive voice response (IVR). These include collecting DTMF digits and playing and recording multimedia content.

MSCML fills the need for IVR and conference control with requests and responses over a SIP transport. VoiceXML [5][6] fills the need for IVR with requests and responses over an HTTP transport. This enables developers to use whatever model fits their needs best.

In general, a media server offers services to SIP UACs, such as Application Servers, Feature Servers, and Media Gateway Controllers. See the IPCC Reference Architecture [21] for definitions of these terms. It is unlikely, but not prohibited, for end-user SIP UACs to have a direct signaling relationship with a media server. The term "client" is used in this document to refer generically to an entity that interacts with the media server using SIP and MSCML.

The media server can potentially fulfill the role of the Media Resource Function (MRF) in the IP Multimedia Subsystem (IMS) as described by 3GPP. MSCML and RFC 4240 [3], upon which MSCML builds, are specifically focused on the Media resource (Mr) interface which supports interactions between application logic and the MRF.

RFC 4722 describes a working framework and protocol with which there is considerable implementation experience. Application developers and service providers have created several MSCML-based services since the availability of the initial version in 2001. This experience is highly relevant to the ongoing work of the IETF, particularly the SIP, SIPPING, MMUSIC, and XCON work groups, the IMS work in 3GPP, and the CCXML [7] work in the Voice Browser Work Group of the W3C.

It is critically important to emphasize that the goal of MSCML is to provide an application interface that follows the SIP, HTTP, and XML development paradigm to foster easier and more rapid application deployment. This goal is reflected in MSCML in two ways.

First, the programming model is that of peer to peer rather than master-slave. Importantly, this allows the media server to be used simultaneously for multiple applications rather than be tied to a single point of control. It also enables standard SIP mechanisms to be used for media server location and load balancing. Second, MSCML defines constructs and primitives that are meaningful at the application level to ensure that programmers are not distracted by unnecessary complexity. For example, the mixing resource operates on constructs such as conferences and call participants rather than directly on individual media streams.

The MSCML paradigm is important to the developer community, in that developers and operators conceptually write applications about calls, conferences, and call legs. For the majority of developers and applications this approach significantly simplifies and speeds development.

As mentioned above, MSCML payloads may be carried in either SIP INVITE or INFO requests. The initial INVITE, which creates an enhanced conference, MAY include an MSCML payload. A subsequent INVITE to the same Request-URI joins a participant leg to the conference. This INVITE MAY include an MSCML payload. The initial INVITE that establishes an IVR session MUST NOT include an MSCML payload. The client sends all mid-call MSCML payloads for conferencing and IVR via SIP INFO requests.

SIP INVITE requests that contain both MSCML and Session Description Protocol (SDP) body parts are used frequently in conferencing scenarios. Therefore, the media server MUST support message bodies with the MIME type "multipart/mixed" in SIP INVITE requests. The media server transports MSCML responses in the final response to the SIP INVITE containing the matching MSCML request or in a SIP INFO message. The only allowable final response to a SIP INFO containing a message body is a 200 OK, per RFC 2976 [20]. Therefore, if the client sends the MSCML request via SIP INFO, the media server responds with the MSCML response in a separate INFO request. In general, these responses are asynchronous in nature and require a separate transaction due to timing considerations.

There has been considerable debate on the use of the SIP INFO method for any purpose. The experience of the MSCML authors is that MSCML would not have been possible without it. At the time the first MSCML specification was published, the first SIP Event Notification draft had just been submitted as an individual submission. At that time, there was no mechanism to link SUBSCRIBE/NOTIFY to an existing dialog. This prevented its use in MSCML, since all events occurred in an INVITE-established dialog. And while SUBSCRIBE/NOTIFY was well suited for reporting conference events, its semantics seemed inappropriate for modifying a participant leg or conference setting where the

only "event" was the success or failure of the request. Lastly, since SIP INFO was an established RFC, most SIP stack implementations supported it at that time. There have been few, if any, interoperability issues as a result.

In order to guarantee interoperability with this specification, as well as with SIP User Agents that are unaware of MSCML, SIP UACs that wish to use MSCML services MUST specify a service indicator that supports MSCML in the initial INVITE. RFC 4240 [3] defines the service indicator "conf", which MUST be used for MSCML conferencing applications. The service indicator "ivr" MUST be used for MSCML interactive voice response applications. In this specification, only "conf" and "ivr" are described.

The media server MUST support moving the call between services through sending the media server a BYE on the existing dialog and establishing a new dialog with an INVITE to the desired service. Media servers SHOULD support moving between services without requiring modification of the previously established SDP parameters. This is achieved by sending a re-INVITE on the existing dialog in which the Request-URI is modified to specify the new service desired by the client. This eliminates the need for the client to send an INVITE to the caller or gateway to establish new SDP parameters.

The media server, as a SIP UAS, MUST respond appropriately to an INVITE that contains an MSCML body. If MSCML is not supported, the media server MUST generate a 415 final response and include a list of the supported content types in the response per RFC 3261 [2]. The media server MUST also advertise its support of MSCML in responses to OPTIONS requests, by including "application/mediaservercontrol+xml" as a supported content type in an Accept header. This alleviates the major issues with using INFO for the transport of application data; namely, the User Agent's proper interpretation of what is, by design, an opaque message request.

6.5 Internet-drafts

6.5.1 SIP Interface to VoiceXML Media Services

The Internet Draft entitled "SIP Interface to VoiceXML Media Services" [4] (previously known as ‘draft-burke’) provides a detailed specification of how VoiceXML 2.0/2.1 media services are invoked using SIP. The specification is currently in the IETF RFC Editors’ queue and is expected to become an Informational RFC shortly.

The specification leverages the Request-URI mechanism in RFC4240 [3] for identifying dialog media services: the user part is fixed as ‘dialog’ and a ‘voicexml’ parameter identifies the URI of the initial document to fetch. Other parameters include HTTP control parameters (e.g. document caching and method) and user-defined data parameters (see below). The Request-URI can be used in SIP INVITE and REFER messages to initiate a VoiceXML session; for example:

> sip: dialog@ms.example.com; voicexml=http://as.example.com/promptandcollect.vxml; maxage=3600; maxstale=0;

The specification details the relationship between SIP signaling and VoiceXML session invocation and termination, error behavior and handling special character in parameters. Detailed signaling and media information about the connection are made available to invoked VoiceXML session through its ‘session.connection’ object. The VoiceXML script can then access basic to/from/redirect and request-uri information as well as SIP INVITE/REFER headers and SDP information (including type, direction and format of each negotiated media stream).

draft-ietf-mediactrl-vxml [4] pays special attention to many area of the relationship between SIP and Voice XML including passing data to/from the AS, media support and outbound calling.

Firstly, data interaction between the AS and VoiceXML session can occur at invocation, mid-call and at termination. The AS specifies data to be injected into the VoiceXML session using user-defined parameters on the initial Request-URI. This data is then exposed in the VoiceXML script allowing the AS to configure specific VoiceXML behavior. For example, if the AS sends an INVITE with the Request-URI

sip: dialog@ms.example.com; voicexml=http://as.example.com/promptandcollect.vxml; prompt=http://as.example.com/prompt1.wav; iterations=10; max-digits=5

then this information can be accessed in the VoiceXML script so the audio prompt can be specified as

<audio expr="session.connection.protocol.sip.requesturi[‘prompt’]"/>

where the expression evaluates to http://as.example.com/prompt1.wav. Similarly, information about the number of attempts to collect input, and the size of the input, can also be set dynamically in the script.

Data collected by the VoiceXML script can be returned to the AS mid-call using HTTP methods. In VoiceXML 2.0, this can be achieved using the <submit> or <subdialog> elements. In VoiceXML 2.1, data can be efficiently sent to the AS using the <script> or <data> elements; in return, the AS can pass new data back to the VoiceXML application. For example, to return the digits collected in a prompt and collect script,

```
<field name="digits">
 <filled>
  <data name="newdata" src="http://as.example.com" namelist="digits"/>
 </filled>
</field>
```

Where the digits collected from user are send to the AS at "http://as.example.com" and in return the AS sends an XML document identified as "newdata". The VoiceXML script could then access this new data and its behavior affected; for example, the data could instruct the VoiceXML script to terminate or continue collecting digits.

Data can also be returned to the AS at the end of the session using SIP or HTTP methods. When data is specified in the namelist of VoiceXML <exit> or <disconnect> elements, the session is terminated and namelist data is returned to the AS in the body of a SIP BYE (or 200 response). Alternatively, data can be returned using the mid-call methods above before it terminates the call or, when the call is terminated by an incoming BYE, during 'post-disconnect' final processing described in VoiceXML.

Secondly, draft-ietf-mediactrl-vxml address media support including dialog preparation, early media and codec support. When the initial INVITE establishes a media-less SIP dialog, the VoiceXML session is prepared (initial document fetched, etc) but not executed until a re-INVITE establishes a media session. The VoiceXML MRFC may also support early media by sending a 183 Session Progress provisional response to the initial INVITE. On media codecs, a VoiceXML MRFC must support codecs and RTP formats which correspond to those mandated in the VoiceXML 2.0/2.1 (G.711 alaw/mulaw); other codecs may be supported. If video is supported by the VoiceXML MFRC, then it must support H.263 baseline and should support AMR.

Finally, outbound calling is an optional feature of VoiceXML 2.0/2.1 which draft-ietf-mediactrl-vxml addresses by specifying how the various types of call transfer must be addressed. For 'blind' transfer (caller is transfer away from the MRFC to a new destination), a REFER message is sent on the original SIP dialog. For 'bridge' transfer (caller is connected to a new callee but the MRFC still receives the caller's media streams), the VoiceXML MRFC establishes a new outbound SIP dialog with a callee and then connects its media streams with the original caller's. Consultation transfer – conceptually similar to blind transfer but the caller isn't dropped if the transfer attempt is unsuccessful – is implemented similar to 'bridge' transfer except that the INVITE contains a Replaces header so as to replace the connection between the caller and MRFC with a connection between the caller and the callee. Note that the specification states the AS should insert its own URI into Record-Route header of the initial INVITE so that it remains in the signaling path when outbound calls are initiated by the MRFC.

The clear benefit of draft-ietf-mediactrl-vxml is that it addresses limitations in RFC4240 with respect to the functionality and interoperability of the VoiceXML dialog service. It does so by explicitly specifying how VoiceXML and SIP interact, including many error cases. It also clearly specifies various methods for the AS to send data to the VoiceXML session and for the VoiceXML session to return data to the AS. This enables the AS and MRFC to exchange data not just at service invocation, but throughout the lifecycle of the media session. Finally, it also details how VoiceXML's optional transfer capability can be treated, assuming an MRFC is able to initiate outbound calls. If an MRFC is not able to initiate outbound calls, then its approach on this topic is not viable. Instead, either the VoiceXML MRFC requests the AS (or CSCF) to generate outbound calls; or VoiceXML transfer capability is not supported in its 3GPP profile.

6.5.2 Media Server Control Requirements Draft

The IETF is has established a working group for Media Control. This activity is focusing on the interface between an Application Server and a Media Server, via SIP and XML based markup languages. There are also prospects that the results of this work could be adapted by the 3GPP for control of an MFRC. The document draft-ietf-mediactrl-requirements [22] (previously known as the "Dolly draft") offers a draft set of requirements for such an interface. In particular, the document addresses the need for an XML-based protocol that can be used to control a variety of different media sessions, which may include Interactive Voice Response and Enhanced Conferencing Control. The document also anticipates that various types of media can be addressed via these controls, including voice, tones and video. The

draft-ietf-mediactrl-requirements builds upon a great deal of industry experience with mid-call XML being carried via SIP. Details on this Internet Draft, entitled "Media Server Control Protocol Requirements", draft-ietf-mediactrl-requirements [22], follow.

The draft-ietf-mediactrl-requirements is a document that provides a set of requirements for controlling media servers as part of the IETF Mediactrl activity. In this working group, there is an interest to work on a protocol that will enable one physical entity that includes the media policy server, notification server and the focus to interact with one or more physical entities that serves as mixer or media server.

The draft-ietf-mediactrl-requirements presents the requirements for such a protocol. It addresses all phases and aspects of media handling in an enhanced conferencing service including announcements and IVR functionality.

The Media Server work uses when appropriate and expands on the terminology introduced in RFC 4353 [19] on the SIP conferencing framework and the XCON conferencing framework. The following additional terms are defined for use within the Media Server work.

Application Server (AS) - The application server includes the conference policy server, the focus and the conference notification server as defined in the SIP conferencing framework [19].

Media Server (MS) - The media server includes the mixer as defined in the SIP conferencing framework [19]. The media server sources media streams for announcements and it processes media streams for functions like DTMF detection and transcoding. The media server may also record media streams for supporting IVR functions like announcing participants.

Notification - A notification is used when there is a need to report event related information from the MS to the AS.

Request - A request is sent from the controlling entity, such as an Application Server, to another resource, such as a Media Server, asking that a particular type of operation be executed.

The document goes on to define a set of Media Control requirements and some operational requirements.

Examples of specified Media Control requirements include:

> REQ-MCP-01 - There MUST be a requirement for a control protocol that will enable one or more Application Servers to control a media server.
>
> REQ-MCP-02 - The protocol MUST be independent from the transport protocol.
>
> REQ-MCP-03 - The protocol MUST use a reliable transport protocol.
>
> REQ-MCP-04 - The application scope of the protocol shall include Enhanced Conferencing Control and Interactive Voice Response. Though the protocol enables these services, the functionality is invoked through other mechanisms.
>
> REQ-MCP-05 - The protocol will utilize an XML markup language.
>
> REQ-MCP-07 - Media types that are supported in the context of the applications shall include audio, tones, text and video.

6.5.3 SIP control framework and packages

The SIP control framework draft-ietf-mediactrl-sip-control-framework [9] instantiates the protocol model with a dedicated control channel based on TCP/SCTP over which XML messages are passed. SIP control framework packages describe specific XML messages for multimedia IVR and conferencing functionality.

In the SIP control framework SIP is used by the AS as a rendezvous protocol for negotiating a media session with the MRFC using the Session Description Protocol (SDP). This approach leverages COMEDIA (RFC 4145) [11] so that the SDPs describe the establishment of a TCP (or SCTP) channel. The COMEDIA (RFC 4145) [11] approach is well established and used in draft-ietf-simple-message-sessions [12] and draft-ietf-speechsc-mrcpv2 [13]. Using a dedicated channel for exchanging messages between the AS and MRFC addresses the problems of using SIP INFO for message exchange (see subclause 4.3.2). This approach also provides an explicit mechanism for discovery and establishment of the control channel.

The control framework approach itself does not define the content of messages transported by the dedicated control channel. Instead the framework defines a mechanism that provides strict requirements on how the control framework

can be used. Techniques similar to the SIP event framework (RFC 3265) are used when creating extensions to the control framework. The control framework introduces the concept of 'Control Packages'. This also provides a mechanism to explicitly identify the capabilities of MRFCs: different MRFCs can support different packages. Packages also facilitate future extensions to MRFC functionality.

IETF working drafts proposals on media server protocol include the following packages as relevant for MRFCs:

- Basic Interactive Voice Response (IVR) Control Package draft-boulton-ivr-control-package [16]: This provides lightweight messages for simple IVR interactions. This control package uses parameterized dialog templates for playing announcement, prompt and collects and prompt and record IVR functions without the need to implement a full VoiceXML solution.

- VoiceXML Interactive Voice Response (IVR) Control Package draft-boulton-ivr-vxml-control-package [17]: This package extends the basic IVR control package with support for VoiceXML. Note that this package does not support VoiceXML's optional call transfer functionality. Use of VoiceXML in control messages covers the IVR functions of the MRF and allows simple as well as complex interactive behavior to be defined in scripts. Existing VoiceXML applications (e.g. voice mail, prepaid, portals, self-service applications) can be easily and rapidly adopted within a 3GPP IMS context with minimal application recoding.

- Conference Control Package draft-boulton-conference-control-package [18]: This package allows for the creation, manipulation and termination of a conference mix. Users, explicitly represented by SIP dialog parameters, can be introduced, moved and removed from an existing conference mix.

- Advanced Conference Control Package (in development): This package provides advanced conferencing capabilities including video conference layout and manipulation, nested conferences, etc.

In terms of architecture, this model uses the existing MRFC reference points together with one additional reference point: a Cr reference point to directly transport media control messages between the AS and MRFC and to allow the MRFC to fetch resources (see subclause 4.3.2.1). The framework also allows the AS to establish multiple dedicated control channels towards the MRFC; it could for example use one channel per MRFC, one channel per session, or other configurations suitable for high availability deployments.

6.6 Others

7 Requirements for a media server control protocol

7.1 Introduction

The present section lists the requirements identified by the conclusion of the studies in the previous sections along with other identified requirements for a media server control protocol.

7.2 Multimedia services' media control requirements

The high level functionality of the MRFC and MRFP is defined in 3GPP TS 23.228 [33].

A media server control protocol must therefore be able to control the tasks of the MRF (MRFC/MRFP) which are:

- Control the media stream resources.
- Generate of CDRs.
- Control of the bearer on the Mb reference point.
- Mixing of incoming media streams (e.g. for multiple parties).
- Media stream source (for multimedia announcements).
- Media stream processing (e.g. audio transcoding, media analysis).

- Floor Control (i.e. manage access rights to shared resources in a conferencing environment).

The media operations required to be performed by a media server control for the implementation of services are further specified in the following 3GPP specifications.

3GPP TS 23.218 [35] and 3GPP TS 24.229 [36] describes the use of an MRF for:

- A single announcement/tone.
- Ad hoc conference.
- Transcoding.
- Session based messaging conferences including multiple UEs (e.g. multiparty chat conferences). The MRFC/MRFP being an option to using an IMS AS to receive messages from conference participants and to distribute messages to all or some of the participants.

3GPP TS 23.333 [37] describes the use of an MRF for:

- Playing tones.
- Playing of audio announcements, including announcement variants (for Date/Day/Month, Time, Digits, Money and Integers).
- Playing of multimedia files.
- Audio and multimedia recording.
- DTMF collection.
- Text to Speech.
- Audio and video transcoding.
- Automatic Speech Recognition.
- Audio and video conferencing with a floor control policy and possible conference maximum size.

3GPP TS 24.147 [24] describes the use of an MRF for:

- Conference mixing, as described in RFC 4353 [19].
- Conference initiation and termination.
- Floor control.

3GPP TS 24.247 [38] describes the possible use (depending on the functional AS/MRFC split) of an MRF for:

- Page-mode messaging.
- Session-mode messaging conferences.

The 3GPP 29.198 series specifications and 3GPP 29.199 series specifications include the following media resource functions:

- Play announcement.
- Execution of VoiceXML scripts.
- Text to Speech.
- Automated Speech Recognition.
- DTMF detection.
- Audio and multimedia conferencing.

There are other media functions which are used today in multimedia applications and which fall under the high-level definition of MRFC/MRFP tasks. A media server control protocol should be extensible and be able to handle the complexity introduced by this type of media function in the future:

- Audio mixing including gain control, N-loudest mixing, forced mix inclusion, whispering/coaching, side-bar/split/merge conference, active speaker notification and DTMF clamping.
- Dynamic video frame rate, bit rate or picture size adaptation per output multimedia stream.
- Media insertion (audio, video, text, picture, logo, avatar or background/ambiance) in a multimedia stream.
- Multimedia mixing including the above 3 functions, different video layout with different video and audio switching capabilities (for example video switching based on audio activity).
- Mixing and media insertion of incoming RTSP streams.
- Multicast media stream output.
- Media stream content caching/hosting/serving.
- VCR control of multimedia streams (pause, resume, play at offset).

There are also some emerging multimedia applications' media functions that a media server control protocol may need to support long term. Examples being shape/face detection/removal, morphing, music recognition, movement/motion detection and voice transformation/masking.

7.3 Response time requirements

For a quality user experience the reaction to a user's input should be rapid.

For an MRF examples of user input are DTMF input, spoken voice commands, floor control commands and user commands arriving as media control instructions from the AS.

Examples of MRF reactions for media related services are:

- Stopping an ongoing audio announcement.
- Stopping an ongoing video output.
- Confirming the input command execution, including the end of a recording or floor control requests, with a new audio or video stream.
- Confirming the input command execution, with a response sent to the application server (which in turn will relay a reaction to the user in some way).
- Pausing, resuming or jumping to an offset in audio or video output media.
- Switching a video layout in a video conference.

If the reaction is slow the user is likely to repeat his command, which may get wrongly interpreted as a new user input and the service may be perceived as non-responsive and sluggish.

There are no specifications for reaction times for an MRF. Reaction time requirements are often specific to a given end-user service and for an MRF may be different depending on the deployment models used as described in subclauses 5.1 and 5.2. The end user reaction time requirement will also depend on all the other elements involved in its implementation.

Typical response times, however, of a media server or IVR performing similar functions to those above are 100ms or below.

The choice of media server control method or methods should allow a rapid reaction time for a quality end user experience.

One property of the delegation model, described in subclause 4.3.1.4, is that the reaction time is reduced by MRFC treatment of the user event when the user event and its reaction arrive from and are sent over the bearer interface (Mb).

7.4 Packaging, registration and extensibility requirements

The media server control method should support organization, registration and extensibility mechanisms as described in subclause 4.6.

7.5 Charging requirements

7.5.1 General

When considering charging requirements for media control related interfaces, the charging architecture, charging principles and charging data for IM CN subsystem described in TS 32.240 [39] and TS 32.260 [40] shall be followed.

The charging information and authorisation requests, defined in TS 32.260 [40], that are generated from an MRFC are today based on SIP/SDP session information. This allows charging based on SIP session durations, SIP events and the media types negotiated for the session but not for charging based on the media resources used by the MRF.

More advanced charging scenarios may require further charging information and authorisation requests generated from the MRFC for:

- The volume, bandwidth and other properties of the media sent to, from or stored by the MRF.
- The media processing resources used by an application or SIP session (for example speech synthesis, transcoding and media mixing resources).
- Split charging or 3rd party charging – charging different users for the media or resource usage by an application or SIP session. For example, charging differently for the above or separately for each media used (audio, video, session based messages).

The extracts, below, from section 4 of TS 22.115 [42] give charging requirements and high-level principles which are relevant to the above scenarios:

- to allow Network operator to 3rd party supplier (for example Value Added Service Provider) charging;
- to support the shared network architecture so that end users can be appropriately charged for their usage of the shared network, and network sharing partners can be allocated their share of the costs of the shared network resources;
- It shall be possible to charge separately for each type of medium used (for example voice, video, data) in a session and for each service used (for example voice call, streaming video, file download);
- It shall be possible to charge for different levels of QoS applied for and/or allocated during a session for each type of medium or service used;
- It shall be possible to charge a user according to the network resources used. For example, if a large bandwidth is required to use high quality video, the user could be charged accordingly. This is related to charging by QoS;
- It shall be possible to charge users flexibly for the use of extra resources (in at least the same network) for all legs of the call. For example, if a video component is added to a voice call the use of extra radio resource at both ends of the call could be paid for by each user in the call or totally by the initiating user;

- It shall be possible for operators to have the option to apply charging mechanisms that are used in GSM/GPRS. For example for duration of a voice call, for the amount of data transmitted (for example for streaming, file download, browsing) and for an event (one-off charge).At present there are no 3GPP service specifications which define the detailed, media related, charging information required.

7.5.2 Mr interface

MRFC shall be able to generate the charging information for charging purposes. The charging related parameters (e.g. ICID, IOI) and charging function addresses received in the incoming initial SIP request will be used by the MRFC for charging purposes.

For detailed information on transporting charging parameters over the Mr interface for SIP aspect, see 3GPP TS 24.229 [36].

7.5.3 Sr and Cr interfaces

The more advanced charging scenarios described in subclause 7.5.1 will require additional charging information to be passed from the AS to the MRFC so that the MRFC can generate the required charging information and authorisation requests.

As this charging information is related specifically to media usage it is appropriate to send this information on the same interface as the media server control operations (rather than adding information in the SIP signalling).

For conferencing the charging information could be defined as part of the media conference policy, as described in subclause 4.4.1.

7.6 Resource Management requirements

One of the tasks for an MRFC, as defined in 3GPP TS 23.228 [33], is to control the resources in MRFPs.

When an MRFC is used for several services (or applications) or by independent application servers these resources need to be isolated between services to avoid one service impacting another service. For example a peak usage in a televoting service should not impact the use of a premium business conferencing service.

It should also be possible to share some resources between services when appropriate.

To isolate or share resources between services it is required that a media server control protocol provides sufficient information so that a request from an AS can be related to the resources required or reserved for a service by the MRFC.

7.7 High Availability requirements

The media server control methods used between the AS and the MRFC should be able to support high availability configurations. There are no specified availability figures for an MRF, availability requirements being network, service and operator specific.

The media server control methods can support high availability configurations via mechanisms such as:

- Using redundant or multiple communication transport channels to avoid single points of failure.
- Using redundant or multiple communication transport channels to reduce the impact of a failure.
- Auditing of an MRFs status for recovery of system and session state after a failure.
- Congestion control to avoid failure due to system overload.
- Backward compatibility to avoid downtime due to maintenance by enabling online software updates.

See subclause 4.3.2 for information on the available options for the dedicated control channel to support high availability.

7.8 QoS control requirements

The media server control method should have a mechanism to inform the AS when the quality of service or resources required for a service are not available or the resources allocated for a service are exceeded. This mechanism should provide AS with information including such as:

- Failure of, or errors, on a media stream, including abnormal reports of lost or dropped packets.
- Incoming media streams including media playing from files which exceed the bandwidth, frame rate or other negotiated or defined resource usage.

The S-CSCF may play a role in managing QoS control, this has not been studied in the present document.

7.9 Security requirements

When considering security requirements for media control related interfaces, the principles and protocols described in TS 32.210 [41] shall be followed.

The media server control interfaces between an AS and an MRFC may cross network boundaries and therefore should be able to cross security domains.

The media server control interfaces should support confidentiality, they may by used to report the user entry of sensitive information such as credit card numbers and PIN numbers.

7.10 Lawful intercept requirements

There are no impacts on this report.

7.11 Priority requirements

There are requirements for some services to be treated by the network with a higher priority, for example, emergency communication and multimedia priority services. Since procedures for media control functionality are part of the overall end-to-end service procedures, priority requirements need to be considered for the media control protocol if the MRFC is involved in the overall prioritized service procedures to ensure the service can be treated with priority end-to-end.

7.12 Other requirements

Annex A:
Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2006-05	CT1#42	C1-060790			Creation		0.0.0
2006-05	CT1#42bis				C1-061323, C1-061324, C1-061325, C1-061326, C1-0613247.	0.0.0	0.1.0
2006-09	CT1#43	C1-061505			C1-061483, C1-061841, C1-061507, C1-061508, C1-061509, C1-061510	0.1.0	0.2.0
2007-02	CT1#45	C1-062195			C1-070572, C1-070573, C1-070574, C1-070124, C1-070125, C1-070627, C1-070576, C1-070628, C1-070629, C1-070579	0.2.0	0.3.0
2007-03	CT#35				V1.0.0 sent for information	0.3.0	1.0.0
2007-05	CT1#46	C1-070750			C1-070751, C1-070891, C1-070753, C1-070892	1.0.0	1.1.0
2007-05	CT1#47	C1-071102			C1-071448, C1-071449, C1-071078, C1-071450, C1-071451, C1-071452, C1-071453, C1-071455	1.1.0	1.2.0
2007-09	CT1#48	C1-071598			C1-072138, C1-072139, C1-072169, C1-072170, C1-072142	1.2.0	1.3.0
2007-10	CT1#49	C1-072346			C1-072566, C1-072573, C1-072574, C1-072688, C1-072689, C1-072690, C1-072691, C1-072692	1.3.0	1.4.0
2007-11	CT1#50	C1-072787			C1-073059, C1-072789, C1-073060, C1-073181, C1-073182, C1-073065	1.4.0	1.5.0
2007-11	CT#38	CP-070775			V2.0.0 sent to CT#38 for approval	1.5.0	2.0.0
2007-12					V8.0.0 created by MCC as CP-070755 w as approved	2.0.0	8.0.0
2008-03	CT#39	CP-080135	0001		Editorial updates : mediactrl drafts	8.0.0	8.1.0
2008-06	CT#40	CP-080357	0002	1	Charging recommendations	8.1.0	8.2.0
2008-06	CT#40	CP-080357	0003	1	Media Resource Broker (MRB) recommendations	8.1.0	8.2.0
2008-06	CT#40	CP-080357	0004	1	TR completion: editorial notes removal	8.1.0	8.2.0
2008-06	CT#40	CP-080357	0005	1	Further discussion on media resource composition	8.1.0	8.2.0