Network Working Group
Internet-Draft
Expires: August 24, 2001

Ott
Kutscher
Meyer
TZI, Universitaet Bremen

February 23, 2001

An Mbus Profile for Call Control
draft-ietf-mmusic-mbus-call-control-00.txt

Status of this Memo

Copyright Notice

Abstract

This document defines an Mbus application profile for call control services. This application profiles is designed to provide the most common basic services of call signaling protocols like SIP[3], H.323/Q.931[4] related to call setup and tear down but also defines a set of optional Mbus commands for supplementary services. The targeted applications include gateway and endpoint decomposition and remote controlling of call signaling engines.

The underlying message passing and addressing mechanisms for the Mbus is defined in the Mbus transport specification[1].

This document is a contribution to the Multiparty Multimedia Session Control (MMUSIC) working group of the Internet Engineering Task Force. Comments are solicited and should be addressed to the

working group's mailing list at confctrl@isi.edu and/or the authors.

Table of Contents

1. Introduction

1.1 Background

The Mbus transport specification[1] defines the transport mechanisms of the Message Bus (Mbus), a local coordination infrastructure that allows message passing between a group of application components.

The Mbus guidelines[2] define a list of conventions for terminology, algorithms and procedures for higher level interaction models that are useful for applications using the Mbus. These conventions are intended as guidelines for designers of Mbus application profiles and Mbus implementations/applications.

This document builds on these two specifications and provides an Mbus application profile for call control services that uses the conventions codified in the Mbus guidelines[2] to specify an Mbus application profile, i.e., a list of Mbus commands and procedures that allow to implement call-control applications.

1.2 Scope of this Document

This document defines a command set and corresponding interactions between application components for basic call control services, such as call setup, call termination. The set of basic call control commands also includes commands for redirecting or forwarding (proxying) call setup requests.

The set of basic call control commands is supplemented by a set of additional commands for supplementary services, such as call hold and call transfer.

In a future version, this document will also specify commands that allow to implement multiparty conferencing.

2. The Call-Control Model

2.1 Overview

The model that this specification is based on is that of a decomposed conferencing system, such as a terminal or gateway. In such a system, there exists a call control engine, for example, a SIP engine, that implements a call signaling protocol, in this case SIP. This call control engine provides the functionality to initiate, manage and terminate call control relations to other endpoints (or gateways).

The call control engine can be viewed as an application component, i.e., it offers certain services to other components that can make use of the call control component and control the call signaling processes. As an application component it is probably designed to be reusable in different application scenarios.

A separate controlling component, termed "controller" in the following sections, implements the application logic and controls one (or more) call control engines using the Mbus commands specified in this document.

The figure below (Figure 1) shows an example of the relation between a controller and a call control engine in an Mbus enables conferencing system.

```
            +---------- Mbus ----------+
            |                          |
            |     +---------------+    |
            |     |               |    |
            |     |   controller  |    |
            |     |               |    |
            |     +---------------+    |
            |             |            |
            |             |            |
            |     +---------------+    |           +---------------+
            |     |     call      |    |    SIP    |     call      |
            |     |    control    |==================|    control    |
            |     |    engine     |    |           |    engine     |
            |     +---------------+    |           +---------------+
            |                          |
            +--------------------------+
```

The scope of this document is the specification of the communication mechanisms between a controller and a call control engine within an Mbus domain, based on the transport mechanisms specified in the Mbus transport specification[1] and based on the interaction schemes defined in the Mbus guidelines[2].

In order to accommodate other call signaling protocols besides SIP, the interactions that are defined here provide a sufficient level of abstraction from concrete call control protocols. This abstraction implies that not every feature of every call control can be provided. The trade-off between generality and functionality/specificity results in a call-control model that

- supports basic, common call control services;
- uses universal addressing schemes for callee addresses and other parameters;
- provides hooks for call control protocol specific extensions, such as optional parameters; and
- separates advanced functionality, such as supplementary services, out into an optional module.

The generality provided should allow for building generic controllers relying on this call-control model that can control call control engine from different protocol domains without having to care about the call control specific details. For an architecture like the one depicted in the figure above (Figure 1) this would allow to replace the SIP call control engine by a H.323 engine without having to change the the implementation of the controller.

2.2 Concepts

This section describes a set of concepts, abstractions and identifiers that are used by the presented call control model. This includes:

identification of calls;

addressing concept for participants and endpoints; and

call state manipulation.

Controlling a call control engine by a controller uses the notion of "a call", which is an abstraction that represents the state of a call control relation that is setup, modified and terminated by means of message exchange between a controller and a call control engine. In order to disambiguate multiple calls that are managed by a system, call identifiers are employed.

Different types of identifiers are used:

Call Identifier: A call identifier is used to identify calls uniquely. In this model, "a call" represents a call control

relation between two endpoints. If an endpoint has call control relation to two other endpoints at the same time, two different call identifiers will be used to disambiguate the call states. The concept of a globally unique call identifier is prevalent in most call signaling protocols as well. For the Mbus call control commands, the call identifiers are generated by the call control engine and are considered opaque values by other components, e.g., a controller. The appearance of the call identifier depends on the call signaling protocoll. See H.225.0[6] and SIP for details.

Call Leg Identifier: Call leg identifiers allow for a more fine grained control of call control relations. A call control engine may try to setup more than one outgoing call at a time in order to establish a call control relation to a participant, e.g., when the call control engine is a component in a forking proxy system. In order to disambiguate the different call legs that are created for a single call, the notion of call leg identifiers is introduced.

Conference Identifier: While a call identifier is used to identify individual call control relations there are also more persistent states, e.g., multi-party conferences. In some models, multi-party conferences can be implemented by creating a full mesh of calls between all participants. The individual calls would then disambiguated with call identifiers while the conference itself is identified by a "conference identifier". In this specification, this identifier is also used to implement call transfer. The transfer of a call is implemented by having the transferor initiate a new call to the transferred-to party, which result in a new call with a new call identifier. In order to be able to identify and track the call it is assigned a persistent conference identifier.

2.3 Basic Services

The provided basic services are:

Call setup: A controller can make a call control engine initiate a new call using its native call signaling protocol. The call control engine will notify the controller of progress events, e.g., when the called party accepts the call. For a called endpoint, the call control engine will signal incoming call events it received via its native call signaling protocol, enabling the controller to react and eventually control the completion of the call setup by accepting the call. See Section 2.3.1 for a detailed discussion of call setup procedures.

Call redirection: After an incoming call has been signaled by the call control engine, the controller can request the call control engine to redirect the call to another endpoint. See Section 2.3.2 for a detailed discussion of call redirection procedures.

Call forwarding/proxying: After an incoming call has been signaled by the call control engine, the controller can request the call control engine to proxy the call to another endpoint. See Section 2.3.3 for a detailed discussion of call forwarding procedures.

Call canceling/rejection: Outgoing and incoming calls can be rejected and cancelled by the controller at any time. See Section 2.3.4 for a detailed discussion of call rejection procedures.

2.3.1 Call Setup

The figure below (Figure 2) provides a schematic visualization of the Mbus communication for setting up and terminating a call. "CS" represents the call signaling protocol. Please refer to Appendix A and to Appendix B for examples that show the mapping of call control specific PDUs to the Mbus commands.

```
                Caller (A)                          Callee (B)


       Controller        Call Control   Call Control     Controller
                           Engine          Engine

         |   call            |              |                   |
         | ----------------> |              |   incoming-call   |
         |                   |              | ----------------> |
         |                   |              |   proceed         |
         |   proceeding      |      CS      | <---------------- |
         | <---------------- |              |                   |
         |                   |     <-->     |   ring            |
         |   ringing         |              | <---------------- |
         | <---------------- |              |                   |
         |                   |              |   accept          |
         |   accepted        |              | <---------------- |
         | <---------------- |              |                   |
         |   connect         |              |                   |
         | ----------------> |              |                   |
         |   connected       |              |   connected       |
         | <---------------- |              | ----------------> |
         |                   |              |                   |

         |   cancel          |              |                   |
         | ----------------> |              |                   |
         |   cancelled       |              |   cancelled       |
         | <---------------- |              | ----------------> |
         |                   |              |                   |
```

The figure above (Figure 2) shows the message flow for a calling party A as well as for a called party B . A's controller initiates the call setup with a "call" message sent to the call control engine. The call control engine would subsequently setup a call using its native call signaling protocol (not shown here in detail). The most important parameters of the "call" message are the address of the callee and a media/capability description to be used for the call.

In case a call control relation with the callee can be established, A's call control engine will notify the controller of call progress indications it received via its call signaling protocol. When B has accepted the call, A's call control engine will notify the controller with a "accepted" message, which must be acknowledged by sending a connect message back to the call control engine. In essence, this mimics a three-way-handshake model, that allows some basic form of call parameters negotiation, as employed by, e.g., SIP[3]. For this purpose, both the "call" and the "accepted" message can be parameterized with media/capability descriptions.

In this example, the call is terminated by A's controller's sending of a "cancel" message to its call control engine, which subsequently terminates the call control relation to B. Both A's and B's call control engine notify their controllers with a "cancelled" message. A "cancel" message can be sent at any stage of a call setup phase in order to terminate the call and cancel the call control relation.

2.3.2 Call Redirection

The figure below (Figure 3) provides a schematic visualization of the Mbus communication for redirecting an incoming call. "CS" represents the call signaling protocol.

```
                 Caller (A)                          Callee (B)


          Controller        Call Control   Call Control     Controller
                              Engine          Engine

            |   call              |              |                    |
            | ----------------->  |              |   incoming-call    |
            |                     |              | -----------------> |
            |                     |              |   redirect         |
            |   redirected        |      CS      | <----------------- |
            | <-----------------  |              |                    |
            |                     |     <-->     |                    |
```

In this example, B's controller decides to redirect the incoming call instead of accepting it. The "redirect" message is parameterized with the address of an alternative contact for the originally called party. This information is transported via the native signaling protocol and reported by A's call control engine to its controller using the "redirected" message.

In order to contact the party that A has been redirected to A's controller would have to setup a new call using the "call" message.

2.3.3 Call Forwarding/Proxying

Call forwarding/proxying is a functionality used for implementing application layer gateways, e.g., proxy servers.

Details TBD

2.3.4 Call Rejection

The figure below (Figure 4) provides a schematic visualization of the Mbus communication for rejecting an incoming call. "CS"

represents the call signaling protocol.

```
                Caller (A)                          Callee (B)

       Controller        Call Control   Call Control     Controller
                           Engine          Engine

         |   call            |              |                   |
         | ----------------> |              |   incoming-call   |
         |                   |              | ----------------> |
         |                   |              |   reject          |
         |   rejected        |      CS      | <---------------- |
         | <---------------- |              |                   |
         |                   |     <-->     |                   |
```

In this example, B's controller decides to reject the incoming call instead of accepting it. The "reject" message is parameterized with a reason code. This information is transported via the native signaling protocol and reported by A's call control engine to its controller using the "rejected" message.

2.4 Supplementary Services

The provided supplementary services are:

Call hold:

Call transfer:

Call waiting:

3. The Mbus Call-Control Profile

3.1 General

This section defines how the call control model described in Section 2 is implemented as an Mbus application profile using the mechanisms defined in the Mbus guidelines[2]. We define Mbus commands that provide the call control services and define the structure and semantics of parameters.

3.1.1 Mbus Parameter Type Definitions

Some commonly used parameter types:

Call reference: In most of the Mbus commands specified below a call reference is used to identify calls. Call control engines can map the call reference to call identifies of their call signaling protocol. The Mbus parameter data type for call references is String and abbreviated as "call-ref" in the specification below. References are created by call (Section 3.5) or incoming-call (Section 3.10) commands. Every newly created call reference MUST be composed of the Mbus address of the creating entity and a second entity specific part in order to ensure uniqueness.

Address: Some commands require the specification of an address (or address list) for users. These addresses are self-contained URIs, that allow to identify the call control protocol domain and the call control domain specific information that is required to setup a call control relation to the specified user. One of following scheme identifiers (see [7] for the definition of the general URI syntax) MUST be used:

sip: for SIP URIs

h323: for H.323 URIs

tel: for telephone call URIs as specified in [8]

The scheme specific part of an address URI MUST contain the protocol specific information that is needed to establish a call control relation.

The Mbus parameter type for an address is called "address" in the specification below. The Mbus type for "address" is String. Address parameters are used in requests to call control engines, that should be able to translate them into native addresses of their corresponding call signaling protocol.

Address list: For some commands, more than one address needs to

passed as a parameter. The type "address-list" is defined as a "list of address" and is used as a parameter type for requests where more than address can be specified.

Logical address: A logical address is an informational address that denominates the user that a caller is trying to call. The logical address is not necessarily identical to the address URI described above. For example, in a SIP-INVITE request the Request-URI may be "sip:123434565@big-company.foo" (which may have been obtained from a location server), whereas the logical address is "sip:support@big-company.foo" (in SIP, this could be the content of a To header field). As the To header field in SIP, the logical address can be augmented by a "display name", that can be presented to a user by a user agent. As an Mbus parameter, the logical address is therefore represented as a list of two elements (both of type string), where the first element is the display name and the second element is the address URI. In the command specification below the type for logical address parameters is called "logical-address".

Status codes: Some of the commands defined below can be parameterized with status codes and reason descriptions that represent error conditions (or other status information). On the Mbus, this information is represented as a list of two strings, where the first element is a numerical status code code and the second element is a textual description. In the command specification below the type for status information parameters is called "status". The details of the status codes are to be defined; they are to be derived from H.323 and SIP.

Media: Some commands have a media parameter list and/or a capability list for media settings for the call. SDP[9] or SDP-ng[10] SHOULD be used for describing session parameters and capabilities. The Mbus parameter type "media" is a pair of (Symbol, Data) where the first element identifies the type of the description language and the second element is the actual description. The following description types SHOULD be used:

* SDP

* SDP-ng

In order to allow for expressing preferences with SDP, some commands use a list of media for media description parameters. In these lists, the order of the media elements (each of which represents a stand alone SDP description) define their relative preference.

   Overview of the parameter data types:

Type name	Mbus type definition	Description
call-ref	string	Call Reference
address	string	Address URI
address-list	list of address	List of URIs
logical-address	pair of string	Logical Address
status	pair of string	Status Information
media	pair of (symbol,data)	Media Information

In the command specification below the type names are used to specify parameter lists.

3.1.2 Mbus Addressing Scheme

The following Mbus address fields SHOULD be used by implementations of the call control commands:

function: Describes the general function of the component.
The value SHOULD be fixed to "call-control" for both, controller and call control engine.

cc-module: Describes the type of the component.
Possible values:

controller: The component is a controller.

engine: The component is a call control engine.

A sample Mbus address for a controller could look like this:

```
(function:call-control cc-module:controller id:123-4@192.168.1.1)
```

A sample Mbus address for a call control engine could look like this:

```
(function:call-control cc-module:engine id:124-4@192.168.1.1)
```

3.1.3 Mbus Control Relation Class

The Mbus guidelines[2] specify different control classes for applications consisting of modules with controller/controllee relations.

Implementations of the call control profile SHOULD implement the control class "tight control", which means, that a controllee (a control engine) can only be controlled by one controller at a time.

A controller MUST therefore take over the control of a call control engine -- using the mbus.register command [2] -- before it can send commands to a call control engine. The command prefix for the call control commands is "conf.call-control". This means, a controller MUST register itself for the "conf.call-control" hierarchy. See Section 3.2 for the default target address that SHOULD be used for event notifications by call control engines that are not yet controlled.

3.2 Mbus Commands

The following Mbus commands can be divided into two classes:

RPCs

Event notifications

RPCs are sent from a controller to a call control engine. All RPCs MUST be supported by call control engines, i.e., they MUST be able to receive and understand them. Where possible, the imperative form has been chosen for RPC command names, e.g., "call" and "cancel".

Event notifications are sent from a call control engine to a controller. All event notifications MUST be supported by controllers, i.e., they MUST be able to receive and understand them. Where possible, the past (or present) participle form has been chosen for names of event notification commands, e.g., "connected" and "proceeding".

The default target address (see the Mbus guidelines[2] for a definition of default target address) is

```
           (function:call-control)
```

3.3 conf.call-control.accept

conf.call-control.accept
RPC

Parameters:

REF: call-ref
Identifies the call the command refers to.

MEDIA-LIST: list of media
A list of media types along with the preferred capability descriptions selected by the local controller. This SHOULD be a strict subset of the media descriptions the calling endpoint has proposed for this particular call.

Optional Parameters: none

Return parameters: none

The following application specific result states are defined:

OK: The parameters are valid and the accept has been sent.

INVALID_REF: The reference is invalid

INVALID_PARAMETER: One or more parameters are invalid. The error description SHOULD provide more detailed information.

An ACCEPT message is sent by the local controller to the call control engine that has indicated an INCOMING CALL (Section 3.10) to indicate acceptance of the call.

3.4 conf.call-control.accepted

conf.call-control.accepted
EVENT NOTIFICATION
default target address: (function:call-control)

Parameters:

REF: call-ref
Identifies the call the command refers to.

Optional Parameters:

LEG: integer
Identifies the leg the command refers to.

The ACCEPTED message is sent by the callers call control engine to the local controller to indicate that the party has accepted the call.

3.5 conf.call-control.call

conf.call-control.call
RPC

Parameters:

REF: call-ref
A unique identifier for the call. This reference MUST be used for all further interactions relating to this between the call control engine and the initiating entity. Call references SHOULD be constructed considering the rules specified in

Section 3.1.1.

CALLER-INFO-LIST: list of string
A list containing caller information that the call control engine should use for this call. The first parameter is the caller's logical address, the second parameter a protocol specific source address (if applicable) and the third the display information (e.g. the real name). The caller-info-list can be empty for scenarios where the call control engine can provide this information itself.

CALLEE: logical-address
The callee's logical address.

DESTINATION-ADDRESS: address-list
An ordered list of URIs -- where the protocol domain is indicated by the scheme prefix of each URI. It is assumed that all these addresses refer to the same user, and only a single call will be established. The order in which the addresses are specified indicates a preference and contacting the target SHOULD be tried in that order.

CALL-TYPE: symbol
Indicates the intention of the call: join a conference (or an n-way call), invite another user into a conference or an n-way call, or create a new call or conference. Possible values are

INVITE-2-PARTY: for an invitation to a 2-party call

Other types are to be defined in future versions of this document.

MEDIA-LIST: list of media
A list of media types along with the preferred capability descriptions to be used for this particular call.

Optional Parameters:

GW-PROXY-LIST: list of string
An ordered list of (ordered) lists identifying proxies or gateways to be used for call setup if they are known. The n-th element in the list is a list of alternative gateways/proxies to be used in the n-th step in the call setup process.

CALL-ID: data
The call-id is a unique call identifier for this call. The type is protocol-dependent, see H.225.0 or SIP for details. If the call-id is not given, the call-control engine MUST

generate one.

CONF-ID: data
The conf-id is a unique conference identifier for this call. The type is protocol-dependent, see H.225.0 or SIP for details. If the conf-id is not given, the call-control engine MUST generate one.

ACTIVE-MC: integer
If different from zero the caller is an active-mc in this call.

TRANFER-REF: call-ref
Indicates that this call setup belongs to a transfer indication with the given reference.

REDIRECT-REF: call-ref
Indicates that this call setup belongs to a forward indication with the given reference.

Return parameters:

CALL-ID: data
The call-id is a unique conference identifier for this call.

CONF-ID: data
The conf-id is a unique conference identifier for this call.

The following application specific result states are defined:

OK: The parameters are valid and a call-setup process has been initiated.

BAD_URI: The given URI for the callee is not supported by this call-control engine.

INCOMPLETE: The given telephone number is incomplete.

NOT_FOUND: The call-control engine cannot reach an endpoint with the given URI.

DUPLICATE_REF: The reference already exists and cannot be used for this call.

INVALID_PARAMETER: One or more parameters are invalid. The error description SHOULD provide more detailed information.

The CALL command is used to setup a new call and is sent by the local controller to the call signaling engine.

3.6 conf.call-control.cancel

conf.call-control.cancel
RPC

Parameters:

REF: call-ref
   Identifies the call the command refers to.

REASON: status
   A list containing the reason of the cancel.

Optional Parameters: none

Return parameters: none

The following application specific result states are defined:

OK: The parameters are valid and the accept has been sent.

INVALID_REF: The reference is invalid

INVALID_PARAMETER: One or more parameters are invalid. The error
   description SHOULD provide more detailed information.

The CANCEL command is sent by the local controller to the call control engine to indicate that the specified call is to be cancelled. It can also be used by the local controller to inform the call control engine that a call has already been terminated by out-of-band communication, e.g. a horizontal conference control protocol.

3.7 conf.call-control.cancelled

conf.call-control.connected
EVENT NOTIFICATION
default target address: (function:call-control)

Parameters:

REF: call-ref
   Identifies the call the command refers to.

REASON: status
   A list containing the reason of the cancel, e.g. normal
   hangup.

Optional Parameters: none

The CANCELLED message is sent by the call control engine to the local controller to indicate that the call was cancelled.

3.8 conf.call-control.connect

```
conf.call-control.connect
RPC

Parameters:

REF: call-ref
   Identifies the call the command refers to.

Optional Parameters:

LEG: integer
   Identifies the specific call leg the command refers to.

Return parameters: none

The following application specific result states are defined:

OK: The parameters are valid and the accept has been sent.

INVALID_REF: The reference is invalid

INVALID_PARAMETER: One or more parameters are invalid. The error
   description SHOULD provide more detailed information.
```

The CONNECT message is sent by a the local controller to the call control engine to establish the call.

3.9 conf.call-control.connected

```
conf.call-control.connected
EVENT NOTIFICATION
default target address: (function:call-control)

Parameters:

REF: call-ref
   Identifies the call the command refers to.

PEER-ADDRESS: address-list
   Specifies the address of the peer endpoint (or a
   proxy/gateway/gatekeeper hiding its actual identity) that the
   call was finally established with.

MEDIA-LIST: list of media
```

A list of media types along with the capability descriptions that were initially negotiated for this particular call.

Optional Parameters: none

The CONNECTED message is sent by a call control engine to the local controller to indicate that the call was successfully established.

3.10 conf.call-control.incoming-call

```
conf.call-control.incoming-call
EVENT NOTIFICATION
default target address: (function:call-control)
```

Parameters:

REF: call-ref
   A unique identifier for the call, that is created by the call control engine signaling in accordance with the rules specified in Section 3.1.1.

CALLER-ADDRESS: pair of (logical address string)
   The address of the caller. The first list element is the logical address, that may contain a display name. The second list element can be an alternative "real" address (if available) or be an empty string.

CALLEE: logical-address
   Callee address as specified by the caller. For example, this may be the content of a SIP To Header.

CALLEE-ADDRESS-LIST: address-list
   An ordered list of URIs, that are addresses of the callee as specified by the caller. For SIP call control engines, this will be a list with one element, with the first element as the SIP Request-URI.

CALL-TYPE: symbol
   Indicates the intention of the call; similar to the CALL-TYPE of the CALL message.

MEDIA-LIST: list of media
   A list of media types along with the preferred capability descriptions proposed by the calling endpoint to be used for this particular call.

CALL-ID: data
   A unique identifier for this call.

CONF-ID: data
   A unique identifier for this conference.

Optional Parameters:

GW-PROXY-LIST:
   An ordered list of (ordered) lists identifying proxies or gateways to be used for call setup if they are known. Similar to the CALL message.

TRANSFER-REF: call-ref
   Indicates that this incoming call belongs to a call transfer. If a valid reference is given, this call was used for the transfer and will be terminated.

REDIRECT-ADDRESS: media-list
   Indicates that this incoming call was redirected to this address from the address list. This parameter is optional, because not all call signaling protocols can provide the required information.

The INCOMING CALL messages is sent by the call control engine to the local controller to indicate a call request from another endpoint.

3.11 conf.call-control.proceed

conf.call-control.proceed
RPC

Parameters:

REF: call-ref
   Identifies the call the command refers to.

Optional Parameters: none

Return parameters: none

The following application specific result states are defined:

OK: The parameters are valid and the accept has been sent.

INVALID_REF: The reference is invalid

INVALID_PARAMETER: One or more parameters are invalid. The error description SHOULD provide more detailed information.

The PROCEED command is sent by a local controller to a call control engine in order to indicate that the call setup, that has been

signaled with an INCOMING-CALL (Section 3.10) command, is still proceeding. A call control engine should restart its timers for call setup timeouts (if applicable) and translate this command to a protocol specific message, e.g. a SIP-TRYING or Q931-CALL-PROCEEDING message, that is to be sent to the originating party.

3.12 conf.call-control.proceeding

```
conf.call-control.proceeding
EVENT NOTIFICATION
default target address: (function:call-control)

Parameters:

REF: call-ref
   Identifies the call the command refers to.

PEER-ADDRESS: address-list
   Specifies the address of the peer endpoint (or a
   proxy/gateway/gatekeeper hiding its actual identity) that
   sends the proceeding information.

Optional Parameters:

CALL-LEG: integer
   Identifies the leg the command refers to.
```

The PROCEEDING command is sent by a call control engine to a local controller in order to indicate that the call, that has been initiated with a CALL (Section 3.5) command, is still proceeding. The call control engine will usually send this command after it has received an according message in its call control protocol, e.g. a SIP-TRYING or Q931-CALL-PROCEEDING message. The reception of a PROCEEDING command does not imply that a user has already been contacted. It merely expresses that the call setup is still in progress.

3.13 conf.call-control.redirect

```
conf.call-control.redirect
RPC

Parameters:

REF: call-ref
   Identifies the call the command refers to.

CALLEE: logical-address
   An identifier for the new callee.
```

```
      ADDRESS-LIST: address-list
         List of addresses where the call should be redirected to.

      ATTR: symbol
         A symbol with the value "TEMPORARILY" or "PERMANENTLY",
         signaling whether the redirection is temporarily or not.

      REASON: status
         A list containing the reason of the redirection.

      Optional Parameters: none

      Return parameters: none

      The following application specific result states are defined:

      OK: The parameters are valid and the redirected has been send.
         The call is terminated by the call signaling engine.

      INVALID_REF: The reference is invalid

      INVALID_PARAMETER: One or more parameters are invalid. The error
         description SHOULD provide more detailed information.
```

The REDIRECT command is sent by the local controller to the call control engine to indicate that the specified call is to be redirected to another specified address. If the command returns with OK, the call is terminated.

3.14 conf.call-control.redirected

```
      conf.call-control.redirected
      EVENT NOTIFICATION
      default target address: (function:call-control)

      Parameters:

      REF: call-ref
         Identifies the call the command refers to.

      CALLEE logical-address
         A universal personal identifier for the callee as specified by
         the caller. For example, this may be the content of a SIP To
         Header.

      ADDRESS-LIST address-list
         List of addresses where the call has been redirected to.

      ATTR: symbol
```

```
         A symbol with the value "TEMPORARILY" or "PERMANENTLY",
         signaling whether the redirection is temporarily or not.

      REASON: status
         A list containing the reason of the redirect.

      Optional Parameters:

      CALL-LEG: integer
         Identifies the leg the command refers to.
```

The REDIRECTED command is sent by a call control engine to the local controller to indicate that the specified call has been redirected to the specified address. The local controller has to setup the redirected call with a new CALL command (Section 3.5). The old call will be terminated. If the user does not want the call to be redirected a CANCEL (Section 3) message must be send to the signaling engine to terminate the call.

3.15 conf.call-control.reject

```
      conf.call-control.reject
      RPC

      Parameters:

      REF: call-ref
         Identifies the call the command refers to.

      REASON: status
         A list containing the reason of the rejection.

      Optional Parameters: none

      Return parameters: none

      The following application specific result states are defined:

      OK: The parameters are valid and the accept has been sent.

      INVALID_REF: The reference is invalid

      INVALID_PARAMETER: One or more parameters are invalid. The error
         description SHOULD provide more detailed information.
```

A REJECT message is sent by the local controller to the call control engine that has indicated an INCOMING CALL (Section 3.10) to indicate rejection of the call.

3.16 conf.call-control.rejected

```
      conf.call-control.rejected
      EVENT NOTIFICATION
      default target address: (function:call-control)

      Parameters:

      REF: call-ref
         Identifies the call the command refers to.

      REASONS: list of pair of (address-list, status)
         The pair specifies which target address has rejected the call
         for which reason. As several different address lists may have
         been tried explicitly, a list of pairs is returned.

      Optional Parameters: none
```

The REJECTED message is sent by a call control engine to the local controller to indicate that the call was rejected.

3.17 conf.call-control.ring

```
      conf.call-control.ring
      RPC

      Parameters:

      REF: call-ref
         Identifies the call the command refers to.

      ADDRESS-LIST: address-list
         An ordered list of URIs -- where the protocol domain is
         indicated by the scheme prefix of each URI. It is assumed that
         all these addresses refer to the same user, and only a single
         call will be established.

      Optional Parameters:

      WAITING: integer
         If given, the callee is in a conference and it may take a
         while before he is finally able to accept the call. A value
         greater than zero represents the position of the caller in the
         waiting queue.

      Return parameters:

      The following application specific result states are defined:
```

OK: The parameters are valid and the accept has been sent.

INVALID_REF: The reference is invalid

INVALID_PARAMETER: One or more parameters are invalid. The error description SHOULD provide more detailed information.

The RING message is sent by the local controller to the call control engine. RING indicates that the controller is willing to accept the incoming call and is now alerting the user. A gateway or proxy system should translate incoming RINGING (Section 3.18) commands into RING commands that are to be sent to the call control engine the incoming call was received from.

3.18 conf.call-control.ringing

conf.call-control.ringing
EVENT NOTIFICATION
default target address: (function:call-control)

Parameters:

REF: call-ref
A unique identifier for the call.

CALLEE: address-list
An ordered list of addresses of endpoints that have been alerted. It is assumed that all these addresses refer to the same user, and only a single call will be established.

Optional Parameters:

CALL-LEG: integer
Identifies the leg

WAITING: integer
If given, the callee is in a conference it may take a while before he is finally able to accept the call. A value greater than zero represents the position of the caller in the waiting queue.

The RINGING message is sent by the call control engine to the entity it received the corresponding CALL (Section 3.5) message from. RINGING indicates that one or more endpoints have been contacted and are now alerting the user.

4. Asynchronous Status Signaling

The use of mbus.status commands as specified in the Mbus guidelines[2] and a list of status code with descriptions will be provided in a future version of this document.

5. Supplementary Services

5.1 conf.call-control.hold

```
      conf.call-control.hold
      RPC

      Parameters:

      REF: call-ref
         Identifies the call the command refers to.

      Optional Parameters:

      MEDIA-AVAILABLE: integer
         If given, media information will be send during the hold. This
         may be information or music.

      Return parameters: none

      The following application specific result states are defined:

      OK: The parameters are valid and the redirected has been send.
         The call is terminated by the call signaling engine.

      INVALID_REF: The reference is invalid

      INVALID_PARAMETER: One or more parameters are invalid. The error
         description SHOULD provide more detailed information.
```

The HOLD command is sent by the local controller to the call control engine to indicate that the specified call is to be hold. The call can be retrieved with the RETRIEVE command.

5.2 conf.call-control.on-hold

```
      conf.call-control.on-hold
      EVENT NOTIFICATION
      default target address: (function:call-control)

      Parameters:

      REF: call-ref
         Identifies the call the command refers to.

      Optional Parameters:

      MEDIA-AVAILABLE: integer
         If given, media information will be send during the hold. This
```

      may be information or music.

The ON-HOLD command is sent by a call control engine to the local controller to indicate that the specified call has been set on hold.

5.3 conf.call-control.retrieve

```
conf.call-control.retrieve
RPC

Parameters:

REF: call-ref
   Identifies the call the command refers to.

Optional Parameters: none

Return parameters: none

The following application specific result states are defined:

OK: The parameters are valid and the redirected has been send.
   The call is terminated by the call signaling engine.

INVALID_REF: The reference is invalid

INVALID_PARAMETER: One or more parameters are invalid. The error
   description SHOULD provide more detailed information.

NOT_ON_HOLD: The call is not on hold and cannot be retrieved.
```

The RETRIEVE command is sent by the local controller to the call control engine to indicate that the specified call is no longer on hold.

5.4 conf.call-control.retrieved

```
conf.call-control.retrieved
EVENT NOTIFICATION
default target address: (function:call-control)

Parameters:

REF: call-ref
   Identifies the call the command refers to.

Optional Parameters: none
```

The RETRIEVED command is sent by a call control engine to the local

controller to indicate that the specified call, that has been put on hold before, has been retrieved.

5.5 conf.call-control.transfer

```
   conf.call-control.transfer
   RPC

   Parameters:

   REF: call-ref
      Identifies the call the command refers to.

   CALLEE: logical-address
      An identifier for the new callee.

   ADDRESS-LIST: pair of (symbol, address-list)
      The symbol describes the type of the address. Possible types
      are "REFERENCE" if the list contains one mbus reference for
      the transfer, or "URI" if the list contains URIs for blind
      transfer.

   Optional Parameters: none

   Return parameters: none

   The following application specific result states are defined:

   OK: The parameters are valid and the redirected has been send.
      The call is terminated by the call signaling engine.

   INVALID_REF: The reference is invalid

   INVALID_PARAMETER: One or more parameters are invalid. The error
      description SHOULD provide more detailed information.
```

The TRANSFER command is sent by the local controller to the call control engine to indicate that the specified call is to be transfered to another specified address or another existing call. If the command returns with OK, the call is terminated.

5.6 conf.call-control.transfered

```
   conf.call-control.transfered
   EVENT NOTIFICATION
   default target address: (function:call-control)

   Parameters:
```

REF: call-ref
Identifies the call the command refers to.

CALLEE logical-address
An identifier for the callee.

ADDRESS-LIST address-list
List of addresses where the call has been transfered to.

Optional Parameters: none

The TRANSFERED command is sent by a call control engine to the local controller to indicate that the specified call has been transfered to the specified address. The local controller has to setup the new call with a new CALL command (Section 3.5). The old call will be terminated. If the user does not want the call to be redirected a REDIRECT message must be send to the signaling engine to terminate the call.

References

[1] Ott, J., Perkins, C. and D. Kutscher, "A Message Bus for Local Coordination", Internet Draft draft-ietf-mmusic-mbus-03.txt, December 2000.

[2] Kutscher, D., "The Message Bus: Guidelines for Application Profile Writers", Internet Draft draft-ietf-mmusic-mbus-guidelines-00.txt, February 2001.

[3] Handley, , Schulzrinne, H., Schooler, and Rosenberg, "SIP: Session Initiation Protocol", Internet Draft draft-ietf-sip-rfc2543bis-02.txt, November 2000.

[4] ITU-T, "Visual Telephone Systems and Equipment for Local Area Networks with Non-Guaranteed Quality of Service", ITU-T Recommendation H.323, 2000.

[5] ITU-T, "ISDN User-Network Interface Layer 3 Specification For Basic Call Control", ITU-T Recommendation Q.931, 1993.

[6] ITU-T, "Call Signaling Protocols and Media Stream Packetization for Packet Based Multimedia Communications Systems", ITU-T Recommendation H.225.0, 2000.

[7] Berners-Lee, , Fielding, and Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[8] Vaha-Sipila, , "URLs for Telephone Calls", April 2000.

[9] Handley, M. and V. Jacobsen, "SDP: Session Description Protocol", RFC 2327, April 1998.

[10] Kutscher, D., Ott, J. and C. Bormann, "Requirements for Session Description and Capability Negotiation", Internet Draft draft-ietf-mmusic-sdpng-req-00.txt, February 2001.

Authors' Addresses

Joerg Ott
TZI, Universitaet Bremen
Bibliothekstr. 1
Bremen 28359
Germany

Phone: +49.421.201-7028
Fax: +49.421.218-7000
EMail: jo@tzi.org

Dirk Kutscher
TZI, Universitaet Bremen
Bibliothekstr. 1
Bremen 28359
Germany

Phone: +49.421.218-7595
Fax: +49.421.218-7000
EMail: dku@tzi.org

Dirk Meyer
TZI, Universitaet Bremen
Bibliothekstr. 1
Bremen 28359
Germany

Fax: +49.421.218-7000
EMail: dmeyer@tzi.org

Appendix A. SIP Call Flow Example

```
                Caller (A)                          Callee (B)


       Controller        Call Control   Call Control     Controller
                           Engine          Engine

         |   call            |                   |                   |
         | ----------------> |   invite          |                   |
         |                   | ----------------> |   incoming-call   |
         |                   |                   | ----------------> |
         |                   |                   |   proceed         |
         |                   |   100 proceed     | <---------------- |
         |   proceeding      | <---------------- |                   |
         | <---------------- |                   |                   |
         |                   |                   |   ring            |
         |                   |   180 ringing     | <---------------- |
         |   ringing         | <---------------- |                   |
         | <---------------- |                   |                   |
         |                   |                   |   accept          |
         |                   |   200 OK          | <---------------- |
         |   accepted        | <---------------- |                   |
         | <---------------- |                   |                   |
         |   connect         |                   |                   |
         | ----------------> |   ACK             |                   |
         |   connected       | ----------------> |   connected       |
         | <---------------- |                   | ----------------> |
         |                   |                   |                   |

         |   cancel          |                   |                   |
         | ----------------> |   BYE             |                   |
         |   cancelled       | ----------------> |   cancelled       |
         | <---------------  |                   | ----------------> |
```

Appendix B. H.323 Call Flow Example

```
                Caller (A)                          Callee (B)


       Controller        Call Control   Call Control     Controller
                           Engine          Engine

         |   call            |                   |                   |
         | ----------------> |   setup           |                   |
         |                   | ----------------> |   incoming-call   |
         |                   |                   | ----------------> |
         |                   |                   |   proceed         |
         |                   |   call-proceeding | <---------------- |
         |   proceeding      | <---------------- |                   |
         | <---------------- |                   |                   |
         |                   |                   |   ring            |
         |                   |   alerting        | <---------------- |
         |   ringing         | <---------------- |                   |
         | <---------------- |                   |                   |
         |                   |                   |   accept          |
         |                   |   connect         | <---------------- |
         |   accepted        | <---------------- |                   |
         | <---------------- |                   |                   |
         |   connect         |                   |                   |
         | ----------------> |                   |                   |
         |                   |  H.245 open       |                   |
         |                   |  logical channels |                   |
         |   connected       |                   |   connected       |
         | <---------------- |                   | ----------------> |
         |                   |                   |                   |

         |   cancel          |                   |                   |
         | ----------------> | release-complete  |                   |
         |   cancelled       | ----------------> |   cancelled       |
         | <---------------  |                   | ----------------> |
```

Full Copyright Statement

Acknowledgement

Funding for the RFC editor function is currently provided by the Internet Society.