The Message Bus (Mbus):

An Abstract Call Control Model
for Modular SIP Implementations

Jörg Ott  jo@tzi.org
Dirk Kutscher  dku@tzi.org

http://www.mbus.org/
The Issue: Controlling an IP Telephone...

- HTTP?
- SIP?
- MEGACOP?
- ...

© 2001 Jörg Ott / Dirk Kutscher
The Issue: Controlling an IP Telephone...

It’s not just about CTI-style control of a telephone...

...it’s about creating an integrated desk area environment!
Decomposing a Telephone...

IP Telephone

Display

Keypad

Phone Appl.

API
...further...

Diagram showing a structure with labeled components:
- Display
- Keypad
- Phone "logic"
- User settings
- Audio device
- SIP UA
- RTP
...and adding “external” entities
BTW: Mbus and (Telephony) APIs

Application Functionality
- API lib
- Mbus mapping
- Mbus transport

User Application

“Server” Application(s)

Parlay
JAIN
JTAPI
...

Server Functionality
Introducing Mbus (Message Bus)

Mbus is a message-oriented coordination mechanism for component-based systems.

Mbus is not an API…
- but may well be used underneath emerging APIs

… and Mbus is not...
- MEGACO (or another form of wide area control)
- CORBA
- simply yet another RPC mechanism
- …
Mbus Features (1)

- Message-based communication between components
- Simple(!) local group coordination
  - autoconfiguration
  - supports unicast, multicast, broadcast
- Security mechanisms
  - message authentication, encryption (optional)
- Programming language independence
- Capable of spanning multiple hosts (on a link)
- Supports variety of interactions
  - peer-to-peer, client-server, multipoint, ...
- Flexible addressing scheme
Mbus Features (2)

◆ Collection of basic semantics
  – synchronization, entity awareness, robustness

◆ Building blocks for application semantics
  – defined in guidelines documents
    – voting/polling
    – RPC-style
    – transparent carriage of data

◆ Extensibility through profiles
  – to be defined for (classes of) modules and applications
    – e.g. straightforward inclusion of presence/messaging services
Mbus Transport Overview

- Tuple-based addressing
  - (name:value) pairs

- Group communication
  - recipients flexibly chosen by destination address
  - allows arbitrary grouping, semantics-based addressing, ...

- Underlying Transport: UDP (multicast/unicast)

- Mbus-layer Reliability
  - provided for point-to-point only
  - one-to-many is best-effort

- Messages
  - UTF-8 encoding
  - header
  - commands, parameters
Mbus Semantics: Profiles

◆ Define generic building blocks
  – e.g. voting/polling for policy modules

◆ Define a framework for a particular type of application
  – e.g. multimedia communication endpoints
  – addressing scheme to use

◆ Define Mbus “commands” for entities within a framework
  – protocol engines, functional entities, etc.
  – e.g. (G)UI, call/conference control, media channel, audio device

◆ Profiles follow “Mbus guidelines”

◆ Profiles defined: call control, audio, RTP, ...
Sample Message

Auth Info

<MESSAGE DIGEST BASE64 ENCODED>

Header

mbus/1.0 4711 94593893434 R \\n(module:ui media:audio id:1234@10.1.2.3) \\
(module:engine media:audio id:2001@10.1.2.4) ()

Configure RTP

rtp.ssrc (42)
rtp.source.cname (42 "dku@tzi.uni-bremen.de")
rtp.source.name (42 "Dirk Kutscher")

Configure codec

audio.codec ("g711-u")
audio.channel.coding ("redundant" "gsm")
conf.transport.address (42 10.0.1.2 (50000 50001) 127)

Select codec

security.encryption.algorithm ("des")
security.encryption.key ("ABCDEF")
security.encryption.state (1)
Component Discovery and Identification

- Components implement *Mbus application profiles*, sets of:
  - Mbus resource definitions
    - ▼ RPCs
    - ▼ events notifications
    - ▼ properties
  - Naming schemes
    - ▼ Use of Mbus addressing mechanisms
  - Control Relations

- Mbus entities announce their presence on the Mbus:
  - Mbus addresses used to express compliance to application profiles

- Mbus entities enter control relationships with each other:
  - E.g.: A phone controller takes control of a call signaling engine
  - Resources provided by an Mbus entity defined by application profiles it implements
Call Control Profile: Overview

Goal:

- Provide an abstraction from the specifics of a protocol e.g. (which SIP extensions are available)
  - and allow simple applications to be protocol-agnostic

- But still preserve powerful features of individual protocols
  - to support sophisticated applications (such as gateways) as well

Solution:

- Few general commands
  - conf.call-control.xyz

- General commands: basic call control and supplementary services

- Protocol-specific commands
  - conf.call-control.sip.xyz
Call Control Concept

- **Controller/Controllee-Model**
  - Call control engine (e.g. SIP UA)
    - implements call signaling protocol
    - has well-defined Mbus interface (“API”)
  - Controller
    - implements generic call control logic
    - may interact with UI (if applicable), access user preferences, etc.

- **Interactions**
  - RPCs
    - controller requests actions from call control engine
  - Event notifications
    - call control engine reports call control events to controller (and others)

- **Abstract Call Control Identifiers**
  - Call-leg
    - differentiate multiple call-legs of a single call, e.g. for proxies
  - Call identifier
    - identifier of a single point-to-point call
  - Conference identifier
    - identifier of a “cooperation relationship” (may persist across calls)
### Generic Basic Call Control Commands

<table>
<thead>
<tr>
<th>Command issued by actor</th>
<th>Indication received by peer</th>
</tr>
</thead>
<tbody>
<tr>
<td>conf.call-control.call</td>
<td>conf.call-control.incoming-call</td>
</tr>
<tr>
<td>conf.call-control.ring</td>
<td>conf.call-control.ringing</td>
</tr>
<tr>
<td>conf.call-control.proceed</td>
<td>conf.call-control.proceeding</td>
</tr>
<tr>
<td>conf.call-control.accept</td>
<td>conf.call-control.connected</td>
</tr>
<tr>
<td>conf.call-control.reject</td>
<td>conf.call-control.rejected</td>
</tr>
<tr>
<td>conf.call-control.redirect</td>
<td>conf.call-control.redirected</td>
</tr>
<tr>
<td>conf.call-control.forward</td>
<td>conf.call-control.forwarded</td>
</tr>
<tr>
<td>conf.call-control.disconnect</td>
<td>conf.call-control.disconnected</td>
</tr>
<tr>
<td>conf.call-control.caps-set</td>
<td>conf.call-control.caps-indicate</td>
</tr>
</tbody>
</table>
Outline: Mbus Interactions in an Endpoint

From a controller’s perspective:

- Determine which (media) engines are available
- Determine which features (codecs, etc.) they support
- Inform SIP UA about these
- Configure preferences into the SIP UA
- Make SIP UA register with registrar
- Place outgoing / accept incoming calls
- Configure media engines according to SDP
- Possibly re-configure media engines upon re-INVITEs
- Adjust volume, (un)mute, etc. with audio device
Traditional Telephony Services

- Mbus commands for supplementary services
  - Focus on simplicity
  - Mapping to SIP (extensions)
    - as well as H.323/H.450

- Mbus abstractions for
  - Call Hold
  - Unattended Transfer
  - Attended Transfer
  - Call Park and Call Pickup
  - Simple conferencing?
Example: Unattended Transfer

A

- call
- proceeding
- ringing

B

- incoming-call
- ring
- accept
- connected
- transfer (to C)

INVITE
100 Trying
180 Ringing
200 OK
INVITE
REFER
200 OK
INVITE (C)

transferred (to C)
call (C)
Sample Application(s)

- SIP endpoint
- StarGate: SIP - H.323 - ISDN Gateway
- AudioGate: ISDN - Mbone Gateway

All three use the same audio engine (RAT).

- SIP Redirect Server
- Looking into stand-alone telephone sets as well...
- Integration of awareness system ongoing...
Conclusion

◆ **Mbus** is a simple lightweight message-oriented coordination protocol
  - allows decomposing applications into components on a local link
  - offers mechanisms for application *extensibility* and *tailorability*
  - enables module re-use
  - does not prescribe what to do with it

◆ **Abstract call-control model** is a lean API for IP-Telephony
  - small set of generic functions + protocol-specific extensions
  - end systems, gateways, registrars, proxies/redirect servers, ...

◆ **Further information:** [http://www.mbus.org/](http://www.mbus.org/)