INTERNET-DRAFT
Expires: May 1996

Carsten Bormann
Universitaet Bremen
Joerg Ott
TU Berlin
November 1995

Elements of Session Semantics
draft-bormann-mmusic-semantics-00.txt

Status of this memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as ``work in progress.''

To learn the current status of any Internet-Draft, please check the ``1id-abstracts.txt'' listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Abstract

This memo presents a set of object classes that can be used to define the local representation of the state of the conferences (sessions) that the local system is involved in (LASSO: local session state objects). In addition, some initial information is given on how to map this state to similar semantics in the ITU T.120 series of recommendations. We think these object classes might be used as a basis for defining the dynamic state variables visible in the horizontal control protocol used in a session.

This memo is a submission to the IETF MMUSIC working group. Currently, this memo is in ``strawman'' stage. Comments of any kind are positively encouraged and should be addressed to the confctrl@isi.edu mailing list.

1. Introduction

Conferencing systems employ session control protocols to achieve a common understanding of the state of the conference both between systems that support each participant and within each such system. So far, no proposal has been made to the MMUSIC working group that defines the set of information that makes up the dynamic state of a

session in course.

This memo presents an early look at LASSO, local session state objects. A tree structure built out of such objects (called a ``dictionary'' in our system) can be used as the focal point for the local interaction of applications and control agents within a participating system (a ``vertical protocol''). We also think that the object classes defined for LASSO can be used as a basis for defining the dynamic state variables visible in the horizontal control protocol that is run between control agents.

2. Notation

We have chosen an exposition of the LASSO classes in a notation that is deliberately different from notations we have used in MMUSIC in the past. This is to avoid wasting any mental energy over the syntax.

Each LASSO type is defined an an object class. By convention, type names are upper case. Objects of these types have attributes and subtrees:

```
(class CLASS-NAME (attributes ...) (subtree ...))
```

There is no strong distinction between attributes and subtree information, but there is an expectation that in a control protocol all attributes for one object will be obtained at once, while subtrees are explicitly asked for.

This notation allows some simple inheritance:

```
(class (DERIVED BASE1 BASE2) (attributes ...) (subtree ...))
```

Attributes/subtrees can be required, optional or virtual (i.e., their status is defined in a derived class only).

Types can be used in the context "(LIST TYPE)" (zero or more instances in an ordered list) and "(REFERENCE TYPE)" (points to other instance).

Basic types are: BOOLEAN, INTEGER, STRING (human-readable), BYTE-STRING, SYMBOL (like strings, used for comparison with each other, but not particularly meant to be human-readable).

3. Useful Types

This section defines some underlying types that will be used in other types.

3.1. USER

A user is mainly defined by her "cname". It is expected that this is also used as the basis for the authentication process.

```
(class USER                             ; "business card attributes"
       (attributes                      ;   of a person (cf. RTCP)
        (cname STRING required)         ; cabo@informatik.uni-bremen.de
        (name STRING required)          ; Carsten Bormann
        (e-mail STRING optional)        ; cabo@informatik.uni-bremen.de
        (phone (LIST STRING) optional)  ; E.164: +49 421 218 70 24
        (fax (LIST STRING) optional)    ; E.164: +49 421 201 70 27
        (location STRING optional)      ; Bremen, Germany
        (organization STRING optional))) ; Universitaet Bremen
```

3.2. PROTOCOL, PROTOCOL-WITH-PARAMETERS, PARAMETER

A PROTOCOL identifies a specific horizontal protocol available for talking to a specific application. A PROTOCOL-WITH-PARAMETERS adds values for parameters for the protocol (such as sampling rates, resolution, etc.). (Note that it is not quite clear where the boundary between protocols and parameters lies: is "H.261" a parameter of "RTP"? Or is "H.261/RTP" a protocol with parameters such as "CIF"/"QCIF"?)

```
(class PROTOCOL
       (attributes
        (type SYMBOL required)          ; IANA registered
        (id BYTE-STRING required)))     ; as per type

(class (PROTOCOL-WITH-PARAMETERS PROTOCOL)
       (attributes
        (parameters (LIST PARAMETER) required)))

(class PARAMETER
       (attributes
        (type SYMBOL required)          ; specific to protocol type
        (value BYTE-STRING required)))  ; as per type
```

4. Information About the Local User and System

This section defines that part of the LASSO object tree that is not immediately associated with a particular conference.

4.1. PRESENCE

A presence is used as the root of the LASSO object tree. (A presence is a specific user ``logged in'' to a specific system -- this allows for the possibility that the user is logged in more than once.) It points to sets of capabilities/preferences defined by the local host and by the user, the set of running applications relevant to the conferences joined by this presence, and this set of conferences.

```
(class PRESENCE
       (attributes
        (name STRING required)          ; ID of person responsible
        (host STRING required)          ; local host
        (number INTEGER required))      ; number of presence on local host
       (subtree
        (host LOCAL-HOST required)      ; capabilities intrinsic to host
        (user LOCAL-USER required)      ; capabilities specific to user
        (applications (LIST APPLICATION) required) ; locally running apps
        (conferences  (LIST CONFERENCE) required))) ; current conferences
```

4.2. LOCAL-USER

The information locally available about a user is that of the type USER, augmented by a set of applications that this user generally wants to have running, a set of preferences among applications that can handle a particular protocol, and a set of applications this user has installed separately from the per-host information (~/bin style).

```
(class (LOCAL-USER USER)
       (subtree
        (initialize (LIST LOCAL-APPLICATION) required) ; apps always wanted
        (preferences (LIST PREFERENCE) required) ; proto-to-app mappings
        (applications (LIST APPLICATION-DESCRIPTION) required))) ; available

(class PREFERENCE
       (attributes
        (protocols (LIST PROTOCOL-WITH-PARAMETERS) required)
        (application-list (LIST (REFERENCE APPLICATION-DESCRIPTION)) required)
                                    ; applications, most preferred first
        ))

(class APPLICATION-DESCRIPTION
       (attributes
        (name STRING required)          ; "NanoFirm Paragraph against Doors"
        (version STRING required)       ; "3.14159"
        (invocation STRING required)    ; pathname used for invoking
        (parameter-spec PARAMETER-SPEC required) ; .sd.tcl (if H.261 "-pH261")
        (protocols (LIST PROTOCOL) required))) ; protocols this app supports
```

4.3. HOST, LOCAL-HOST, INTERFACE, DEVICE

Information on a host includes its hostname, a set of interfaces available for networking (details to be defined -- e.g., ISDN/H.320 vs. IP vs. both), and a set of devices that could be useful in a multimedia conference (examples are given for cameras and speakers).

```
(class HOST
       (attributes
        (hostname STRING required))     ; /host/hostname == /.host
       (subtree
        (interfaces (LIST INTERFACE) required) ; network attachments
        (devices (LIST DEVICE) required))) ; hardware capabilities
```

The information available locally on a host generally includes a set of descriptions of installed applications.

```
(class LOCAL-HOST HOST)
       (subtree
        (applications (LIST APPLICATION-DESCRIPTION) required))) ; sw caps
```

A host may have a number of network interfaces of various kinds. IP interfaces, in particular, have an address and can be classified as to their approximate bandwidth.

```
(class INTERFACE
       (attributes
        (type SYMBOL virtual)))

(class IP-INTERFACE
       (attributes
        (type SYMBOL "IP")
        (address BYTE-STRING required)
        (bandwidth BANDWIDTH optional)))

(class BANDWIDTH
       (attributes
        (type SYMBOL virtual)))
```

A host may have a number of devices of interest in a conference setting. Each of the possible kinds of devices has some specific attributes, of which two examples are given here.

```
(class DEVICE
       (attributes
        (type STRING virtual)           ; defined in derived class
        (system-handle STRING required))) ; "/dev/audio0" "host:0" etc.

(class (SPEAKERS DEVICE)
       (attributes
        (type STRING "audio-out")
        (stereo BOOLEAN required)))

(class (CAMERA DEVICE)
       (attributes
        (type STRING "video-in")
        (color BOOLEAN required)))
```

4.4. APPLICATION

Information on a running application includes information on the kind of application and information on how to address this application.

```
(class (APPLICATION APPLICATION-DESCRIPTION APPLICATION-ADDRESS))
```

5. Information about Conferences

5.1. CONFERENCE

Conferences are the focal point of LASSO. They have a human-readable name, an ID, various attributes, are based on a conference profile, have a defined list of members, involve resources (defined key-value relationships), can be described by global capabilities, and in particular are structured into groups of media agents (applications).

```
(class CONFERENCE
       (attributes
        (name STRING required)
        (conference-id BYTE-STRING required)
        (me (REFERENCE MEMBER) optional) ; required when I'm in the conference
        (locked BOOLEAN optional)       ; no additional members allowed
        (listed BOOLEAN optional)       ; conference is being announced
        (security CONFERENCE-SECURITY optional))
       (subtree
        (profile PROFILE required)
        (members (LIST MEMBER) required)
        (resources (LIST RESOURCE) required)
        (global-caps (LIST CAPABILITY) required)
        (appl-groups (LIST APPLICATION-GROUP) required)))
```

A resource is something that can be allocated, identified by a name

(or a set of other attributes). It is intended that the fact that a resource was allocated is known throughout the conference (e.g. for locking a resource). RESOURCE itself is a virtual class (which is extended per kind of usage).

```
(class RESOURCE
       (attributes
        (type SYMBOL virtual)
        (key BYTE-STRING required)))
```

The conference security attributes need to be defined.

```
(class CONFERENCE-SECURITY
       (attributes
        (type SYMBOL virtual)))
```

5.2. PROFILE

The conference profile is the static, reusable part of the conference state (generally defined prior to a conference).

```
(class PROFILE
       (attributes
        (name STRING required)
        (conductible BOOLEAN default false)
        (terminates-when-empty BOOLEAN default true)
        (security CONFERENCE-SECURITY optional))
                                    ; privilege lists etc.
       (subtree
        (agenda (LIST STRING) optional)
        (documents (LIST CONTRIBUTION) optional)
        (roles (LIST ROLE) required)
        (rules (LIST RULE) required)))
```

Roles classify the participants of a conference. Each role is associated with a set of permissions and a set of users that can take on that role.

```
(class ROLE
       (attributes
        (name SYMBOL required)
        (description STRING required)   ; human-readable
        (permissions (LIST PERMISSION) optional)
        (members (LIST USER) optional)))

(class PERMISSION
       (attributes
```

```
        (name SYMBOL required)))        ; symbolic name for permission
```

Rules define the way specific decisions are made. A CONDUCTOR-RULE means that the chairperson of a conference simply can make that particular decision, while a QUORUM-RULE requires a vote from the members (at least from those wielding a particular permission).

```
(class RULE
       (attributes
        (name SYMBOL required)
        (voting-type SYMBOL virtual)))

(class (CONDUCTOR-RULE RULE)
       (attributes
        (voting-type SYMBOL fixed "conductor")))

(class (QUORUM-RULE RULE)
       (attributes
        (voting-type SYMBOL fixed "majority")
        (required-permission SYMBOL optional) ; e.g. "rep" so guests can't vote
        (acceptance-percentage RATIONAL required)))
```

5.3. APPLICATION-GROUP

An APPLICATION-GROUP contains information for a set of peer media agents.

```
(class APPLICATION-GROUP
       (attributes
        (protocol PROTOCOL-WITH-PARAMETERS required)
        (context (LIST PARAMETER) required))    ; common application parameters
       (subtree
        (per-member (LIST APP-PER-MEMBER) required)))

(class APP-PER-MEMBER
       (attributes
        (member (REFERENCE MEMBER) required)
        (contact APPLICATION-ADDRESS required)
        (caps (LIST CAPABILITY) required)))

(class APPLICATION-ADDRESS
       (attributes
        (type SYMBOL virtual)))

(class (INTERNET-APPLICATION-ADDRESS APPLICATION-ADDRESS)
       (attributes
        (type SYMBOL fixed "IP/PORT")
        (ip-address BYTE-STRING required)
        (port INTEGER required)))
```

5.4. MEMBER

An object of type MEMBER represents a presence within a conference.

(Note the the current definition has a major shortcoming: a presence may actually represent a set of human beings.)

```
(class (MEMBER USER)
       (attributes
        (note STRING optional)          ; "out to lunch" (cf. RTCP)
        (contributions (LIST CONTRIBUTION) optional)
        (roles (LIST SYMBOL) required)) ; conductor, guest, proponent...
       (subtree
        (host CONFERENCE-HOST required)
        (caps (LIST CAPABILITY) required)))

(class (CAPABILITY PROTOCOL-WITH-PARAMETERS))
; same structure, but additional types allowed...

(class CONTRIBUTION                     ; document "brought to the meeting"
       (attributes
        (name STRING required)          ; id / reference / description
        (uri STRING optional)))         ; obtain where?

(class (CONFERENCE-HOST HOST)
       (attributes
        (type (LIST SYMBOL) required))) ; end-system, mixer, translator, etc.
```

6. Security Considerations

This memo does not define any access control that may be required on elements of the session state. This needs to be done in the context of a conference policy.

7. Authors' Addresses

Carsten Bormann	Joerg Ott
Universitaet Bremen FB3 MZH 5180	Technische Universitaet Berlin FR 6-3
Postfach 330440	Franklinstr. 28/29
D-28334 Bremen	D-10587 Berlin
GERMANY	GERMANY
cabo@informatik.uni-bremen.de	jo@cs.tu-berlin.de

Appendix: T.120 Support

This appendix defines a few classes that could be used to maintain T.120 specific information. This is done by defining additional derived classes that inherit from the classes defined in the main body of this document.

```
(class (T-120-CONFERENCE CONFERENCE)
       (attributes
        (superior-name STRING optional)
        (superior-modifier STRING optional)
        (local-name STRING optional)
        (local-modifier STRING optional)
        (modifier STRING required)
        (domain-name STRING required)))

(class (T120-TOKEN RESOURCE)
       (attributes
        (type SYMBOL fixed t120-token)
        (token INTEGER required)))

(class (T120-CHANNEL RESOURCE)
       (attributes
        (type SYMBOL fixed t120-channel)
        (channel INTEGER required)))

(class (T120-CONFERENCE-HOST CONFERENCE-HOST)
       ; additional types: terminal (= end-system?), mcu, multiport-terminal
       (attributes
        (node-id INTEGER required)      ; GCC user id of this node
        (superior-node-id INTEGER optional) ; GCC user id of superior node
        (alternative-node-id INTEGER optional)
        (node-name STRING optional)     ; "Bremen", equal to location???
        (site-information STRING optional) ; useful stuff about the site
        (management-device BOOLEAN required) ; has mgmt function
        (peripheral-device BOOLEAN required))) ; subordinate to other device???

(class (T120-APPLICATION-ADDRESS APPLICATION-ADDRESS)
       (attributes
        (type SYMBOL fixed "MCS-SMC")
        (single-member-channel INTEGER required)))
```

Appendix: SDP Support

This appendix defines additional attributes of the session state elements based on attributes defined in the SDP protocol. [This needs to be completed.]

An SDP-announced conference has additional attributes, which here are attached to a class derived from CONFERENCE. Some of these attributes may be PROFILE attributes, instead.

```
(class (SDP-CONFERENCE CONFERENCE)
       (attributes
        (address MULTICAST-ADDRESS required)
        ; strictly speaking, this address is only a default for the media
        ; and simply should be required there
        (creator USER required)
        (version NUMBER required)
        (creating-host HOST required)
        (description STRING optional)
        (uri STRING optional)
        (time STRING optional)          ; zero or more times
        (repeat-spec STRING optional)
        (bandwidth BANDWIDTH optional)
        (attributes (LIST PARAMETER) optional)))
```

SDP defines the ``k='' attribute, which is a kind of CONFERENCE-SECURITY.

```
(class (SDP-K-CONFERENCE-SECURITY CONFERENCE-SECURITY)
       (attributes
        (type SYMBOL fixed "FIXED-DES-ENCRYPTION")
        (key BYTE-STRING required)))
```

The conference bandwidth is a special kind of bandwidth value.

```
(class (CT-BANDWIDTH BANDWIDTH)
       (attributes
        (type SYMBOL fixed "CT")
        (value NUMBER required)))
```

Addressing for IP multicasting generally is per-media.

```
(class (MULTICAST-ADDRESS ADDRESS)
       (attributes
        (type SYMBOL virtual)))

(class (IPV4-MULTICAST-ADDRESS ADDRESS)
       (attributes
        (type SYMBOL fixed "IP4")
        (ip BYTE-STRING required)
        (ttl NUMBER required)))

(class (MULTICAST-MEDIA PROTOCOL-WITH-PARAMETERS)
       (attributes
        (address MULTICAST-ADDRESS optional) ; defaults to per-conf address
        (port NUMBER required)))
```