Project Number:	**IST-1999-10077**
Project Title:	 **Adaptive Resource Control for QoS Using an IP-based Layered Architecture**
Deliverable Type:	**PU - public**

Deliverable Number:	**IST-1999-10077-WP2.2-TUD-2204-PU-O/b0**
Contractual Date of Delivery to the CEC:	**June 30, 2002**
Actual Date of Delivery to the CEC:	**June 28, 2002**
Title of Deliverable:	**Description of user applications for the second trial**
Workpackage contributing to the Deliverable:	**WP 2.2**
Nature of the Deliverable:	**O – Other (Specification)**
Editor:	**Falk Fünfstück (TUD)**
Author(s):	**Andreas König (BAG), Haris Tsetsekas (NTU), John Karadimas (QSY), Falk Fünfstück, Anne Thomas (TUD)**

Abstract:	This deliverable D2204 describes the applications offered to the end-users of the second trial. It firstly demonstrates their functionality and typical usage scenarios. Secondly, the deliverable shows the QoS requirements of the applications in detail and how they are handled by Application Profiles.

Keyword List:	AQUILA, IST, Legacy Application, Basic Internet Application, Complex Internet Service, QoS Requirement, Application Profile

Executive Summary

This deliverable describes the user applications of the second trial, their functionality, usage scenarios, and support by the End-user Application Toolkit (EAT).

"This deliverable describes the functionality offered to the user of the second trial. It takes into account experiences from the experimental integration carried out in the first trial phase. It is explained how the use of the generic End-User Application Toolkit makes the production of application software more controllable and economic. Technical details on the End-User Application toolkit are not covered here but in deliverables D2201 and D2203. Target audience for this deliverable are people deciding on the actual external appearance of the trial to the end-user, including representatives of the end-user community." [Technical Annex]

D2204 is the continuation of D2202. Whereas D2202 describes the user applications of the *first* trial which were Basic Internet Applications (BIA), D2204 focuses on the issue of so-called Complex Internet Services (CIS). CIS bind existing BIAs to a new, value-added service with QoS support.

The CIS for the second trial is Mediazine. It bundles several Basic Internet Applications to some extent already known from the first trial: RealSystem, NetMeeting, and the interactive game OIDS.

The CIS Mediazine is the main issue of D2204 and is described in detail (functionality, screen shots, usage scenarios) within this document. Like in D2202, the deliverable also contains the QoS requirements and the appropriate Application Profiles for Mediazine.

Moreover, D2204 also presents SIGMA, based on SIP, which is another application for the second trial. In this manner it is possible to demonstrate further advanced functionality of the EAT for the second trial: The EAT transparently supports standardized signalling protocols such as SIP used by many upcoming Internet applications (VoIP, etc.).

Table of Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Experience/conclusions from the first trial

The applications used in the first trial are so-called Basic Internet Applications (BIA), i.e. usual Internet applications such as Voice over IP (WinSip), video conferencing (NetMeeting), video streaming (Real Player/Server), and multi-user online games (Unreal Tournament, Ultima Online). All these applications are legacy ones and are in fact not QoS-aware, i.e. they cannot request for QoS. They were used as they are; without any modifications [D2202].

The approach to support the first trial applications with QoS was a twofold: First, to allow manual QoS reservations for them at different usage levels via the AQUILA's QoS portal, and second, to build special proxies that retrieve important information for the QoS reservations directly from application's data flow.

Regarding the manual reservation requests, particularly for the regular (usual) reservation mode, the concept of the **Application Profiles** was introduced for the first trial. Application Profiles hide the technical details of QoS reservations from the end-user and allow the mapping between the network/protocol level and the end-user level of QoS. Moreover, they keep the technical characteristics of the applications persistent so that they can be used to easily establish a number of QoS sessions.

One outcome of the first trial is that the Application Profiles can support a lot of different Internet applications with different service components (Audio, Video, Data) and different QoS requirements. To prove that, five Application Profiles for the five above mentioned applications were developed and tested in the first trial. The profiles cover all for QoS requests necessary information at different abstraction levels, and they can easily be modified and/or extended. Furthermore, new profiles can be created and validated on the basis of the Application Profile's Document Type Definition (DTD).

However, the first trial profiles are a proprietary AQUILA solution. For example, the profiles contained only AQUILA specific QoS parameters but not generic ones that would allow a mapping to other QoS architectures. Also the creation of new profiles that may rely on the same service components and/or codecs as the existing ones could be supported in a better way. The Application Profiles for the second trial are therefore designed to be more generic and extensible (see appendix). For further information, please, refer to [D2203] and [D1203].

Also the application support by proxies was successfully tested during the first trial: On the one hand, the basic approach is to implement several **Application Proxies** (or protocol gateways) relying on one *proxy framework* that provides basic functionality for communication with the EAT.

On the other hand, an H.323 proxy was implemented and tested with NetMeeting that filters for QoS reservations important information such as IP addresses and port numbers from the H.323 traffic. However, this proxy is not able to establish itself a QoS session but only to give

the required information on demand. (The end-user still have to establish the QoS session by using the EAT GUI.)

For the second trial, an additional proxy for the Session Initiation Protocol (SIP) has been realised without any major modifications on the framework. SIP is actually used by many new session based applications, such as Voice over IP (e.g. WinSip). And the Proxy is now able to itself establish a QoS reservation by asking the EAT when it detects a new SIP session [D2203].

1.2 New ideas for the second trial applications

One main innovation of the second trial is the definition of so-called Complex Internet Services (CIS). A CIS is a web platform integrating, binding and presenting BIAs under one roof and supplying those BIAs with QoS via the AQUILA QoS API.

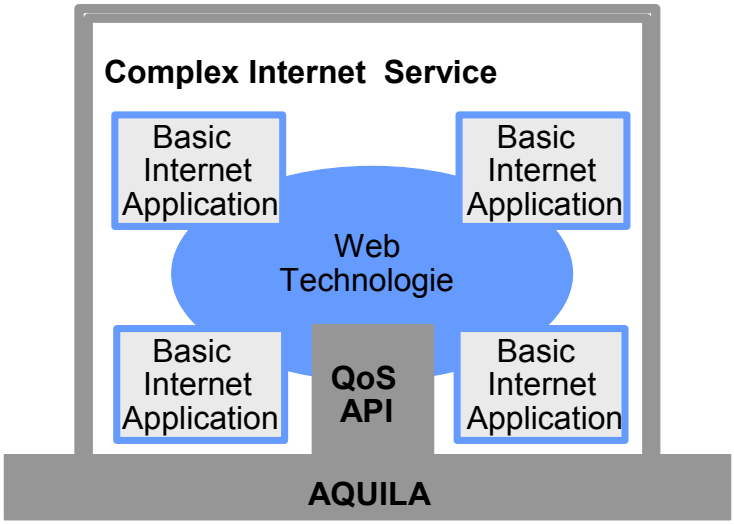

Figure 1-1: Complex Internet Service and AQUILA

One aim of the second trial consists in building such a CIS named **Mediazine**. Mediazine integrates a couple of BIAs and implements the AQUILA QoS API provided by the EAT. As for the first trial an Application Profile corresponds to each integrated BIA. The web platform overtakes the role and functions of the QoS portal of the first trial. The benefit of such a solution is that the BIA is integrated with the QoS component in a platform and that the applications do not run simultaneously to the QoS portal at the beginning of the session. Mediazine will be used on the one hand to further test the concept of the Application Profiles particularly at end-user level, and on the other hand to test three different network services, namely, beside best effort, PVBR (for video conferencing), PMM (for audio and video streaming), and PMC (for interactive game laying).

Another new application for the second trial is the SIP User Agent (SIP UA) **SIGMA** (SIP-based IntelliGent Multimedia Application). SIGMA is an existing application for audio and

video conferencing based on SIP (see chapter 2.2). During the trial, SIGMA will be used as an example application together with the SIP Proxy in order to test, on the one hand, the initiation of QoS requests by protocol gateways for session protocols such as SIP. On the other hand, SIGMA allows the testing of two different network services in parallel, namely PCBR for audio communication, and PVBR for video communication.

1.3 Content of this deliverable

The major part of this deliverable is the description of the user applications for the second trial, which can be found in the next chapter 2. Here, the purposes, the functionality, some significant screenshots, as well as some sample usage scenarios are given in order to introduce the applications and how they rely on QoS.

After that, a more detailed analysis of the QoS requirements of the applications and their components forms the chapter 3, which is called “Application Profiles” in order to point out the relevance of the given information (for example the QoS options, the traffic values, etc.) for creating flexible and reusable profiles. The Application Profiles themselves are given in XML in the appendix.

The deliverable finishes with a summary of achieved status as well as an outlook to further research and development activities.

The appendix contains the actual EAT API, Application Profile document type definition, and the Application Profiles for the second trial.

2 User Applications for the Second Trial

The following chapter introduces the user applications of the second trial. For both, Mediazine and SIGMA, the overall purposes regarding the second trial are given at first, whereas the functionality of the application and their components, which may rely on QoS, are described at second. Finally, some useful usage scenarios for real end-users are presented in order to support the user trials.

2.1 Mediazine

2.1.1 Purpose

Mediazine is a Complex Internet Service (CIS) which provides high quality video and audio content to movie and music fans. It combines several Basic Internet Applications (BIA) to offer the user a value-added service. Mediazine is an example for a commercial relevant broadband Internet service; this original CIS has been tested in an enhanced version by the Bertelsmann Broadband Group in two cable networks in Germany. It offers video and audio on demand over a content delivery network. It includes furthermore an accounting and billing system to test user's acceptance and willingness to pay for such a service.

The AQUILA version of Mediazine implements important functions for the project: To test several network services, the CIS is extended by the communication tool NetMeeting (for the PVBR class) and the game "OIDS" (for the PMC class). The PMM class can be tested by using the video and audio streaming tool RealSystem (with RealPlayer on the client side and RealServer on the server side). To offer an attractive and useable service to the test persons, some additional features (e.g. email, text chat, white board, file transfer) are available, which will not necessarily be tested with AQUILA network services. All other features of the original CIS by Bertelsmann are not needed and therefore disabled.

The minimum requirement for the usage of Mediazine is a 56 kBit/s modem, because the usage of a video and audio platform for broadband content is not sufficient with lower connection speed. To test the network services with different connection speed and bandwidth, the video streams are prepared in three different qualities:

- a 45kBit stream for the usage via modem or (dual) ISDN connection,
- a 384kBit stream for usage via cable modem or DSL connection, and
- a 1024kBit stream for usage via LAN connection.

The audio streams are also prepared in three different qualities:

- a 48kBit stream for usage via modem or ISDN connection,
- a 112kBit stream for usage via dual ISDN connection, and

- a 256kBit stream for usage via cable modem, DSL or LAN connection.

It is possible to test several network services at one time, e.g. a user can be listening to music or watching a movie while talking with another user about the music or the video via the integrated NetMeeting tool; that might be an interesting scenario to test the interaction of different network services.

Connection capacity overload is not intercepted by Mediazine, e.g. listening to music, using the communication tool and downloading a file at the same time with a 56kBit/s modem connection.

2.1.2 Functionality/Components

Mediazine is only usable with Microsoft Windows and Microsoft Explorer 5.x and higher, because some DirectX components are needed. RealPlayer version 6.x and higher and NetMeeting version 3.x and higher have to be installed. It is recommended to use a computer with an 800 MHz processor or a faster machine with 128 MB RAM or more. An excellent video quality for the high bandwidth streams cannot be guaranteed with slower computers.

Mediazine is designed for a resolution of 1024x768 pixels for the main browser window. Other resolutions does not work perfectly, because the resolutions of the videos are not adapted to other resolutions of the main browser window.

2.1.2.1 The main multimedia components

After starting Mediazine, a login screen is presented with two input lines for the user name and the password. Also a “Login” button is shown. The button starts the GUI of Mediazine, if user name and password are correct. Otherwise an error message is printed on the login screen (see Figure 2-1).

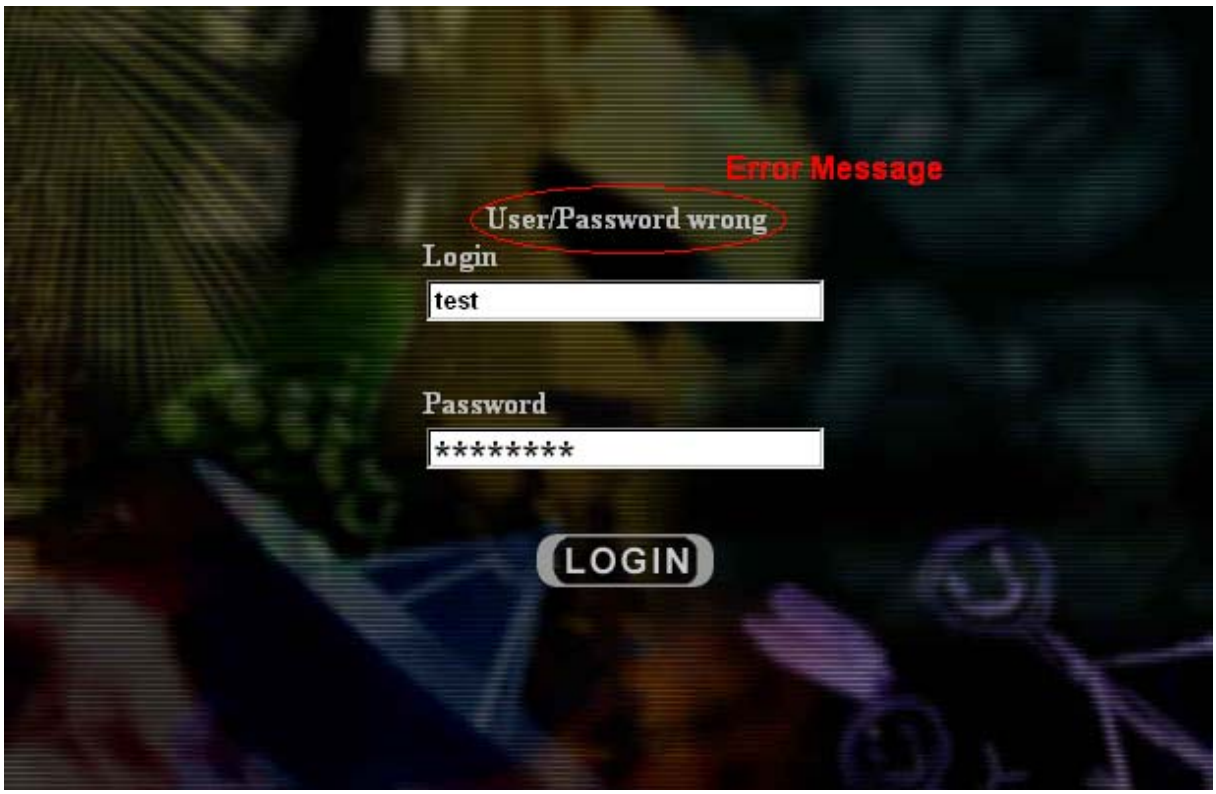

Figure 2-1: Login Screen

The following figure shows the GUI of Mediazine, which is divided into four parts.

Figure 2-2: Mediazine GUI – Start Screen

The four parts (red marked) of the GUI are:

Part A) The control line

The following table presents all buttons of the control line with their functions:

	The “Back” button goes one step back in the history of the used Mediazine pages, if possible.
	The “Forward” button goes one step forward in the history of the used Mediazine pages, if possible.
	The “Home” button goes to the start screen of the Mediazine GUI after the login.
	The “Set-up” button presents in Part C a set of parameters to configure Mediazine. Language, connection and QoS parameters can be set to the user’s preferences. (A description of these features can be found in chapter 2.1.2.2.)
	The “Email” button opens the email front end in Part C. (A description of this front end can be found in chapter 2.1.2.3.)
	The “Logout” button closes the Mediazine GUI and shows the login screen again.
	The “RealControl” panel offers following functions (from left to right) to control a video or audio stream “Start”, “Break”, “Stop”, “Rewind”, “Jump”, “Fast Forward”, “Sound off/on”. (A detailed description can be found in the “Help” menu of the RealPlayer.)

Table 2-1: Control Line

Part B) The menu

The menu offers three main categories: “Movies”, “Music” and “Games”. The category “Movies” includes four sub-categories: “Thriller”, “Action”, “Comedy” and “Miscellaneous”. The category “Music” includes five sub-categories: “Classical Music”, “Black Music”, “Music of the 80’s”, “Rock & Pop” and “Trance”. The category “Games” includes no sub-category.

After selecting a (sub) category, a list of available movies, songs or games are presented in Part C. For the movies (see Figure 2-3): A short description of their content is shown as well as “Start Movie” buttons to start the corresponding video streams with the pre-selected quality (to select a quality see chapter 2.1.2.2). The alternative qualities can be selected by links.

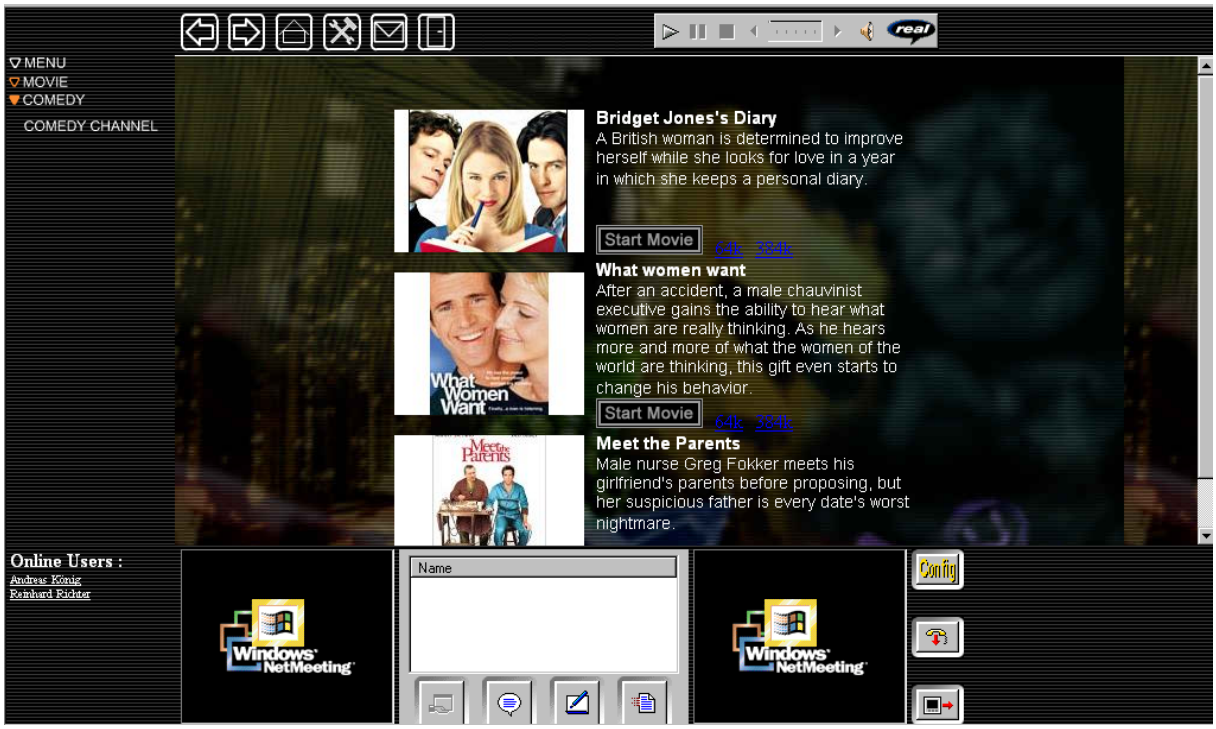

Figure 2-3: Movie Information Screen

For the music: Some play lists (see Figure 2-4) with example pieces of music for the lists are presented corresponding to the selected sub-category. The “Play Music” button for the pre-selected quality of the audio stream is highlighted with a yellow frame. The alternative qualities can be selected by two not highlighted buttons. After selecting a play list, all pieces of music of the selected list are shown also in Part C (see Figure 2-5). Each piece of music can be loaded separately by clicking on the corresponding link, and it is played after preparing it by the audio streaming server.

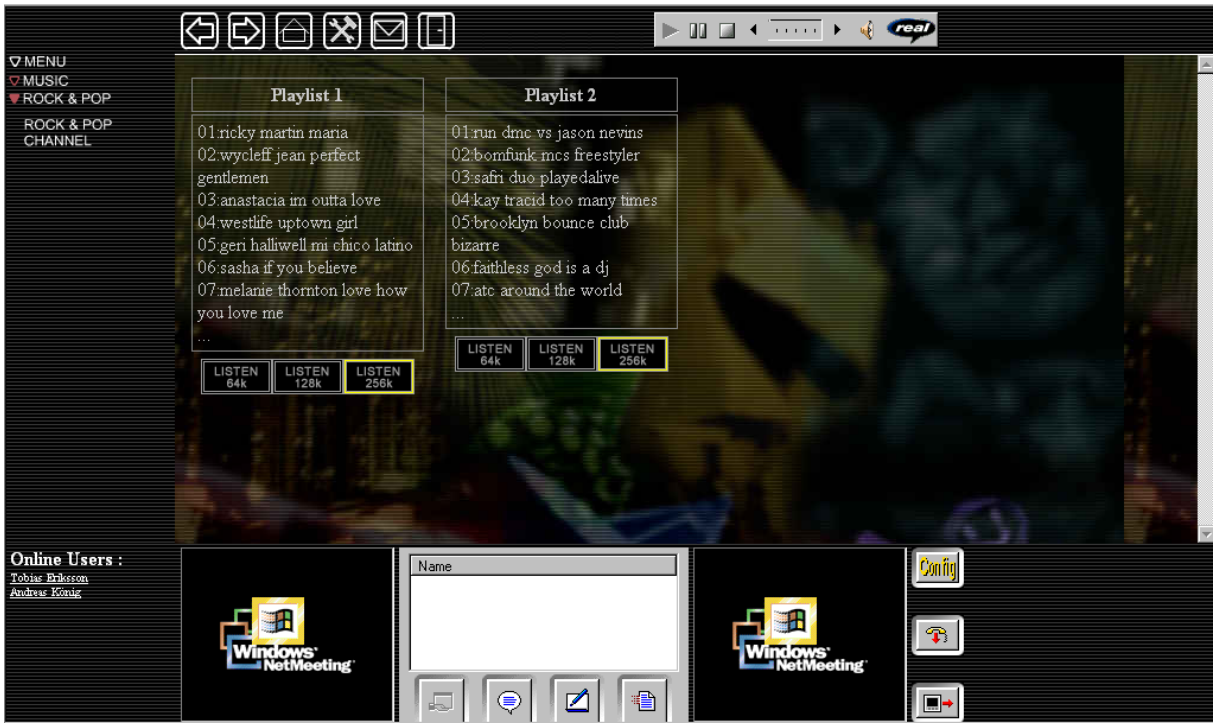

Figure 2-4: Play List Screen

For the Games: Each game with the exception of OIDS is loaded into Part C (see Figure 2-6). A short instruction about the usage of the game is shown. For OIDS (see Figure 2-7) a short description of the game with a link for downloading a MS Word instruction manual is presented. The OIDS game is loaded into a separate window. Note: Only the game OIDS is qualified to test the PMC class. All other games are stand-alone applets and run locally on the client's machine.

Part C) The movie, music and game area

The movies, songs and games are presented in this area. The RealPlayer must be installed to watch movies and listen music. After selecting a movie or a song, the RealPlayer loads data into a buffer (default buffer time: 10 seconds). When the buffer is completely loaded, the selected movie or song starts automatically (see example movie in Figure 2-8).

The RealPlayer control is available in the control line at Part B. For the games, a short load time occurs before the game applet can start.

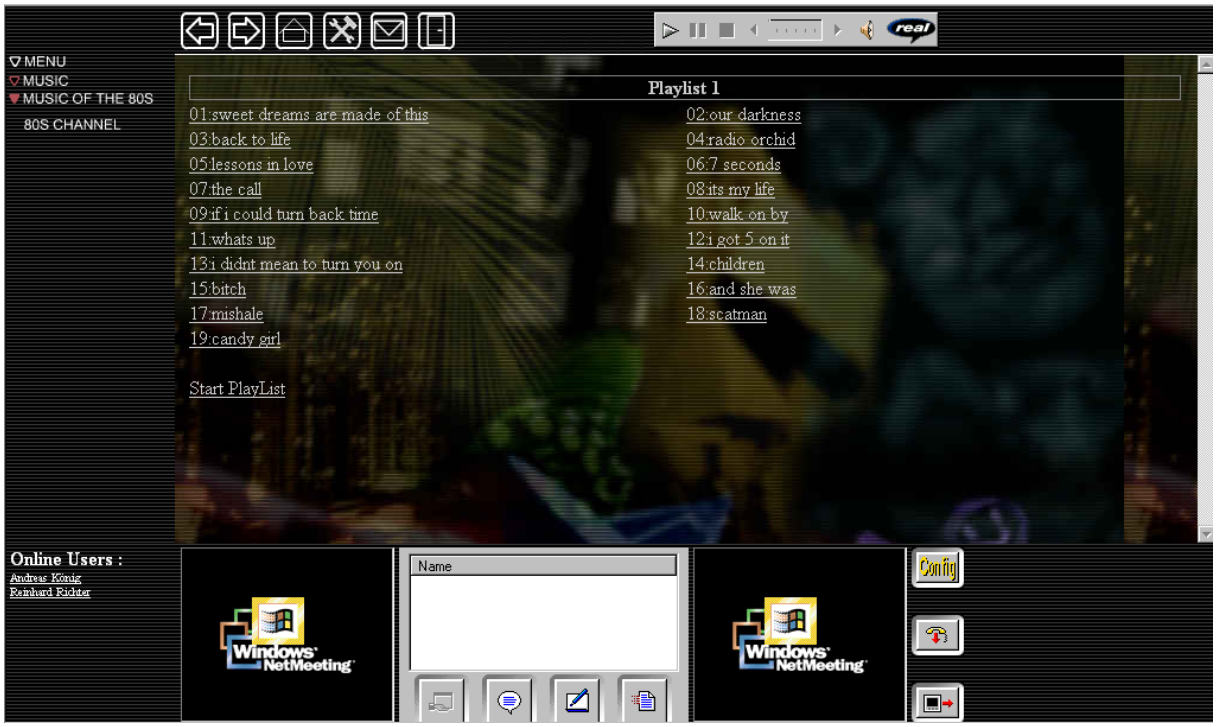

Figure 2-5: Play List Detail Screen

Note: The language of RealPlayer depends on the installed version on the client's host.

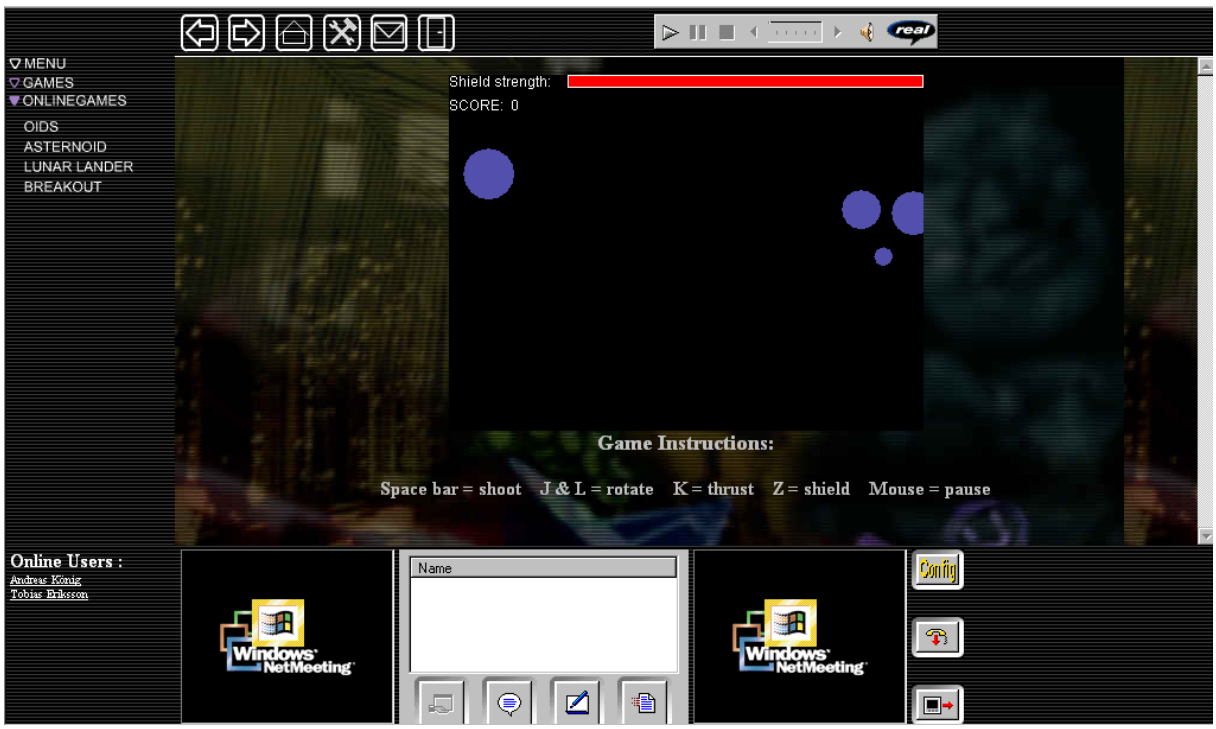

Figure 2-6: Game Screen

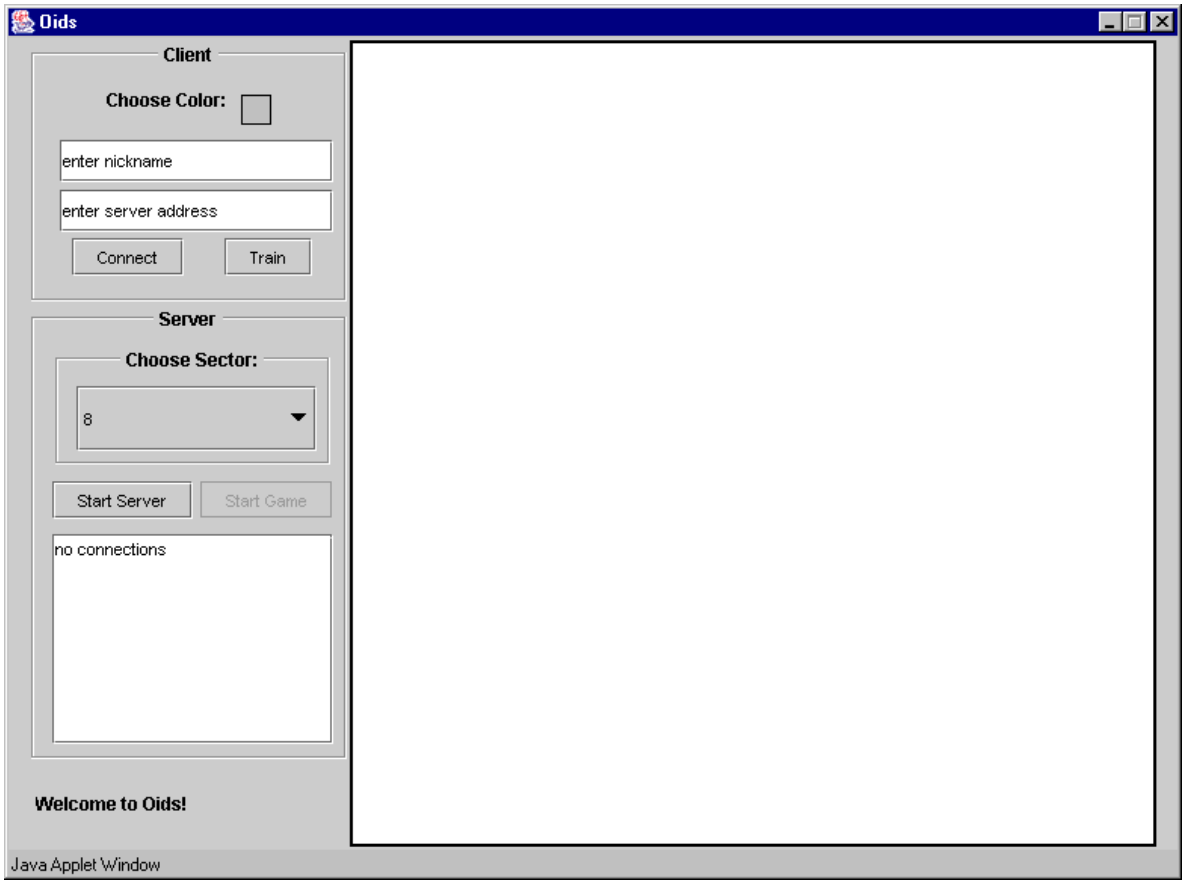

Figure 2-7: The OIDS game

Figure 2-8: Movie Scene

Part D) The communication area

The communication area presents from the left side to the right side a list of available Mediazine users (a), the window for the video of the user (b), the control panel (c), the window for the video of the meeting partner (d), a button area with a "Configuration", a "Call" and an "End Call" button (e) (see Figure 2-9). The parts (b) to (d) as well as the "Call" and "End Call" buttons are based on Microsoft NetMeeting. A description can be found under:

http://www.microsoft.com/windows/NetMeeting
http://www.devx.com/netmeeting
http://www.meetingbywire.com

After clicking on one of the user names within (a), a connection to this user is established. When using Mediazine the first time, the "Configuration" button should be used to configure the NetMeeting tool for Mediazine.

Note: The language of NetMeeting depends on the installed version on the client side.

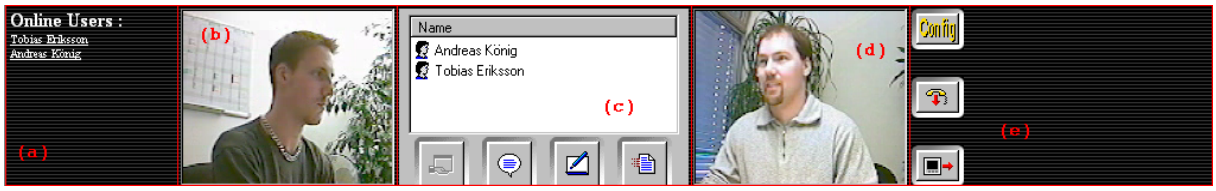

Figure 2-9: Communication Area

All texts within the Mediazine GUI as well as within the login and error message screens are in English. Only the description of the content is available in English and German depending on the configuration set-up of Mediazine. The default language is English. The movies are available in English and also in German.

2.1.2.2 Configuration of Mediazine

After selecting the "Set-up" button (see chapter 2.1.2, Part A), the configuration menu (see Figure 2-10) is presented with values adjusted to the user settings coming from the user profile. These settings are used to establish the needed QoS connection(s) after login into Mediazine.

The menu offers the possibility to set the language of the content description as well as the language of the movies to English or German (a). The user is also able to set-up his/her kind of connection to the Mediazine server (b). Allowed values are "56K Modem", "ISDN", "Dual ISDN", "Cable modem/DSL" and "LAN". For other kind of connections, a suitable classification has to be elected. Slower connection than 56kBit/s is not supported. It is possible to choose between no QoS support or QoS support on a bronze, silver and gold level (c). The QoS levels have the following meaning:

QoS Level	Basic Internet Application	Quality
Bronze	RealSystem audio	very low quality (48 kBit/s)
	RealSystem video	very low quality (45 kBit/s)
	NetMeeting:	very low quality (14.4 kBit/s or 28.8 kBit/s)
	OIDS game:	QoS support enabled
Silver	RealSystem audio	medium quality (112 kBit/s)
	RealSystem video	medium quality (384 kBit/s)
	NetMeeting:	medium quality (ISDN)
	OIDS game:	QoS support enabled
Gold	RealSystem audio	high quality (256 Bit/s)
	RealSystem video	high quality (1024 kBit/s)
	NetMeeting:	high quality (LAN)
	OIDS game:	QoS support enabled

Table 2-2: Quality Levels

A detailed description of the different QoS levels and their coherency with the Mediazine service can be found in chapter 3.1.

There are two buttons “Apply” and “Reset” (d) which have the following functions: The “Reset” button discards all changes and resets all valued to the personal preferences according to the last saved user profile. The “Apply” button stores all changes in the user’s profile and configured the QoS connections.

Note: For the first usage, the configuration menu is set to the following default values: English language, 56 kBit/s modem connection and no QoS level set.

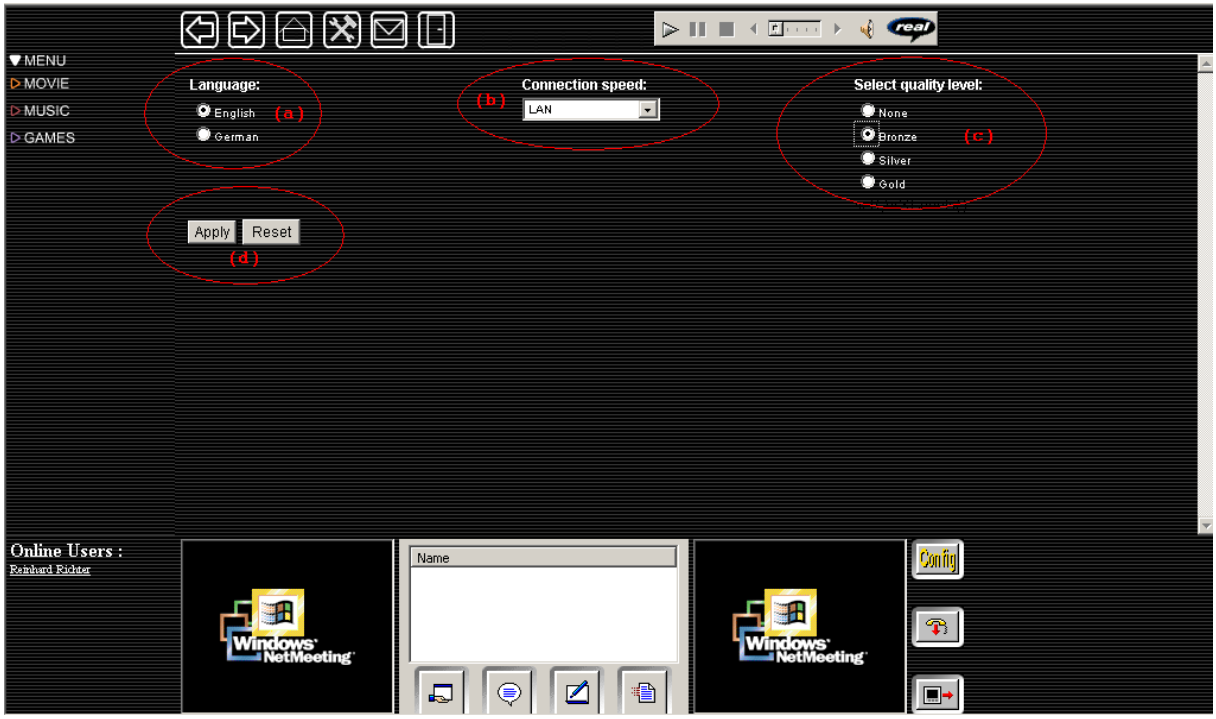

Figure 2-10: Configuration Menu

2.1.2.3 The Email Front End

Figure 2-11 shows the integrated email tool of Mediazine. This tool is a free software tool with the name IMP using Turba. IMP is the Internet Messaging Program (formerly, among other things, the IMAP webMail Program), a Web mail system and a component of the Horde project. IMP is the most widely-deployed component of Horde. IMP offers most of the features which users expect from their conventional mail programs, including attachments, spell-check, address books, multiple folders, and multiple-language support.

Turba is the Horde address book and Contact Management application. It grew out of the need for a more complete address book than the one built in to older versions of IMP. It provides a generic front-end to searching LDAP, SQL, and other contact sources.

Horde is both a piece of software and a project. The Horde Project comprises a set of Web-based productivity, messaging, and project-management applications, each of which is described below. The Horde Framework is a common code-base used by Horde applications, including libraries and a common user interface.

More information can be found under:

http://www.horde.org/imp
http://www.horde.org/faq/admin
http://www.horde.org/papers/kongress2001-imp
http://www.horde.org/papers/oscon2001-horde_tutorial

There is also an online help within the email tool (see Figure 2-11, red box).

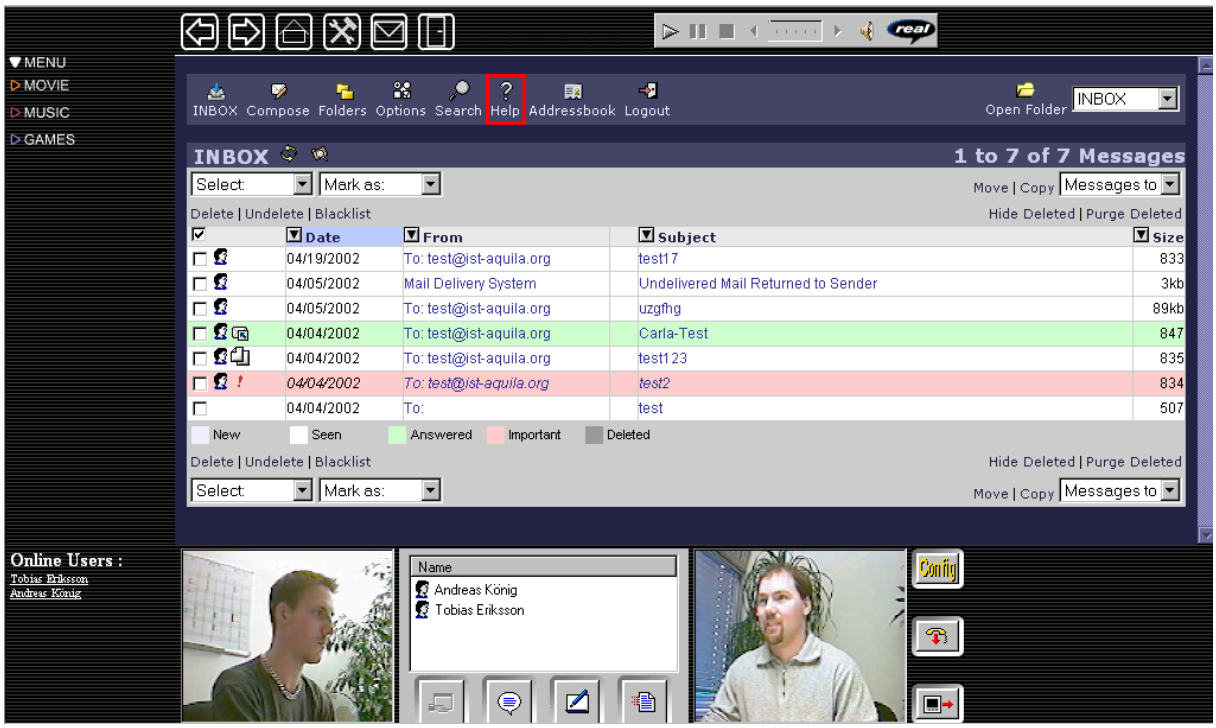

Figure 2-11: Email

2.1.3 Usage scenarios

To see significant differences between QoS enabled and disabled Mediazine components during the trial, it is recommended to use the best quality of the video and audio streams. User tests with lower quality streams are more important for presenting several billing and accounting scenarios for QoS based Complex Internet Services, but that is out of the scope of the AQUILA project.

Note: It is recommended to select movies and scenes for the video streaming service which have a lot of action in them. Fast moving pictures shows the differences in the quality of QoS enabled and disabled streaming services much better than pictures with less movement.

After the first login into Mediazine, the user configures in the “configuration menu“ (see chapter 2.1.2.2) his connection and the preferred language as well as the preferred QoS level (normally “none” at the beginning). The selected values are stored in a user profile, so that later logins do not need a new configuration of Mediazine. There are three usage scenarios in general which are relevant for QoS support (A) and two scenarios without QoS support (B):

A 1. Watching a video

A 2. Listening to music

A 3. Playing a game

B 1. Using the email tool

B 2. Using the video conferencing tool

The scenarios marked with A are described in more detail in the following sub-chapters. The scenarios marked with B are not relevant for the AQUILA trial and are not described in detail. (For more information to these both scenarios see also chapter 2.1.2.3 and chapter 2.1.2.1, Part D).

2.1.3.1 Scenario: Watching a video

The user selects a video after reading the short information of the offered videos (see also chapter, Part B and Part C). If the quality of the video does not fit to his wishes, the user can re-configure the QoS level within the "configuration menu" (see chapter 2.1.2.2) and start the video again by using the "RealControl panel" (see also chapter 2.1.2.1, Part A) or selecting the video again. The user watches the movie until the video ends after approximately 90 minutes.

The user can select another user from the user list within the communication area to talk via a video conference about the watched movie (see also chapter 2.1.2.1, Part D). If the quality of the video conference does not fit to his wishes, the user can also re-configure the QoS level within the "configuration menu".

2.1.3.2 Scenario: Listening to music

The user selects a song or a play list after reading the offered play lists (see also chapter 2.1.2.1, Part B and Part C). If the quality of the music does not correspond to his wishes, the user can re-configure the QoS level within the "configuration menu" (see chapter 2.1.2.2). The user uses the "RealControl" panel (see also chapter 2.1.2.1, Part A) to listen a short track of the song, or the user loads another song by selecting it in the play list. The estimated usage time is one hour.

The user can select another user from the user list within the communication area to talk via a video conference about the listened music (see also chapter 2.1.2.1, Part D). If the quality of the video conference does not fit to his wishes, the user can also re-configure the QoS level within the "configuration menu".

2.1.3.3 Scenario: Playing a game

The user selects a game from the game list after reading the game instructions (see also chapter 2.1.2.1, Part B and Part C). All games except the OIDS game are stand alone games. The estimated play time is 5 to 10 minutes per game.

Only the OIDS game is QoS supported. This game is a multi-player game usable by up to four players. One player has to configure the game as a server and the others as a client. All play-

ers have to select the same play area. A description about the game features and the way to play the game can be found on the instruction screen for OIDS.

If the quality of the connection is not good enough, the user can re-configure the QoS level within the “configuration menu” (see chapter 2.1.2.2).

The user can also select a game partner from the user list within the communication area to talk via a video conference about OIDS (see also chapter 2.1.2.1, Part D). If the quality of the video conference does not fit to his wishes, the user can also re-configure the QoS level within the “configuration menu”.

2.2 SIGMA

2.2.1 Purpose

One of the innovations of the AQUILA project is the introduction of the concept of the Protocol Gateways (Proxies). Protocol Gateways constitute a QoS interface that is provided to the user by the End-User Application Toolkit, in addition to the Graphical User Interfaces (GUIs) and the EAT API.

Protocol Gateways take part in the message exchange of popular signalling protocols, like SIP and H.323, and they initiate a resource reservation procedure in the Resource Control Layer on behalf of the user. More specifically, they detect the establishment of a new session in those protocols, extract all the information that describes this session and forward this information to the EAT.

For the second trial, the Session Initiation Protocol (SIP) is supported with the incorporation of a SIP Proxy into the EAT. The operation of the SIP Proxy, as well as the concept of Protocol Gateways in general, will be tested with the use of a SIP User Agent (SIP UA) application, SIGMA.

2.2.2 Functionality/Components

SIGMA is a SIP User Agent application provided by Siemens. It uses the Session Initiation Protocol (SIP) for the establishment and release of IP telephony sessions, comprising audio as well as video. Apart from the normal SIP operation described in [RFC2543], SIGMA supports also the SIP extension for Resource Management proposed in the IETF draft [Manyfolks]. However, this functionality is not used in AQUILA, since the QoS-related operations are carried out by the SIP Proxy.

SIGMA provides the following configuration options:

- Call configuration: Users may instruct SIGMA to forward the incoming calls to another SIP UA.

- Server configuration: In this property sheet, the IP addresses of the SIP Registrar and the SIP Proxy may be configured. The SIGMA users should insert here only the address of the SIP Server provided by the EAT. Registration functionality is out of scope for the SIP Proxy for the second trial.

- QoS configuration: This sheet in related to the SIP extensions for Resource Management defined in [Manyfolks]. It should be left blank in the AQUILA trial.

- Codec configuration: Allows the configuration of the preferred audio and video codecs to be used in the IP telephony session. Users may alter the codec priority list and may adjust the properties of the video (size, frame rate) and audio (sending frame duration) sent.

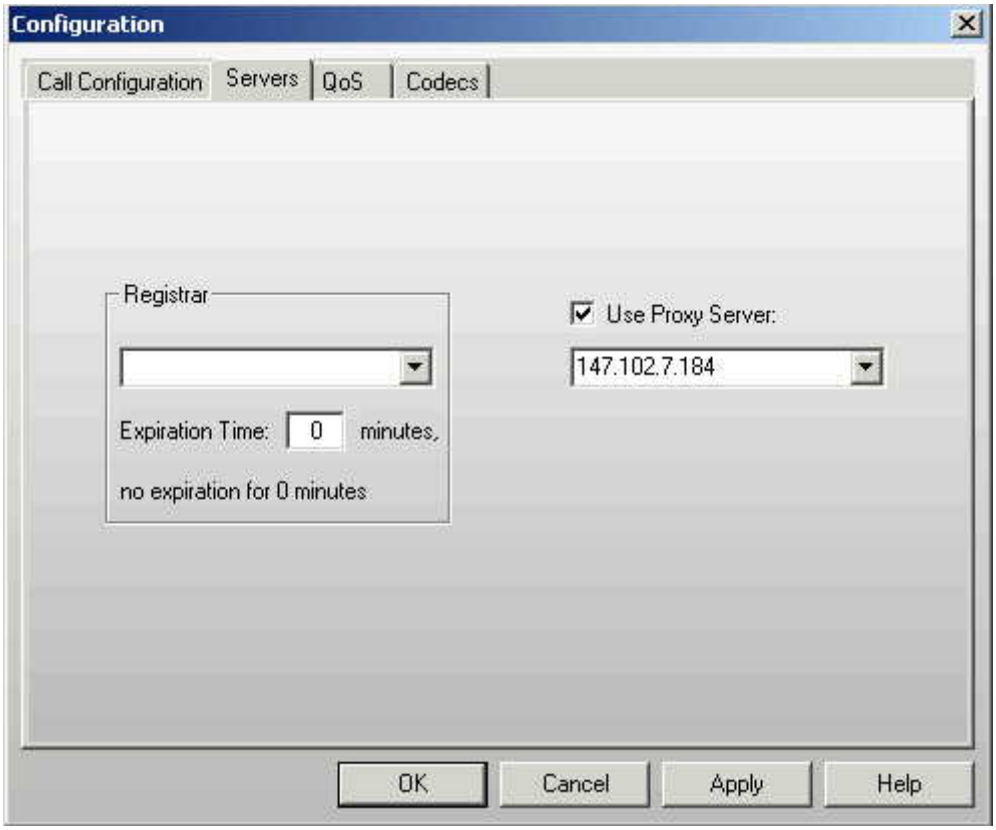

Figure 2-12: Configuration options for the SIP Proxy

2.2.3 Usage scenarios

After SIGMA has been configured properly (basically, the SIP Proxy address has been entered) it may be used for call establishment. During start-up, users are requested to register. Since registration functions are not supported by the SIP Proxy, the SIP addresses of the users entered in this dialog box should have the format: username@host-address and not username@sip-address (for example: htset@147.102.1.1 and not: htset@sip.ntua.gr). After "Accept" is pressed, the main window of SIGMA appears.

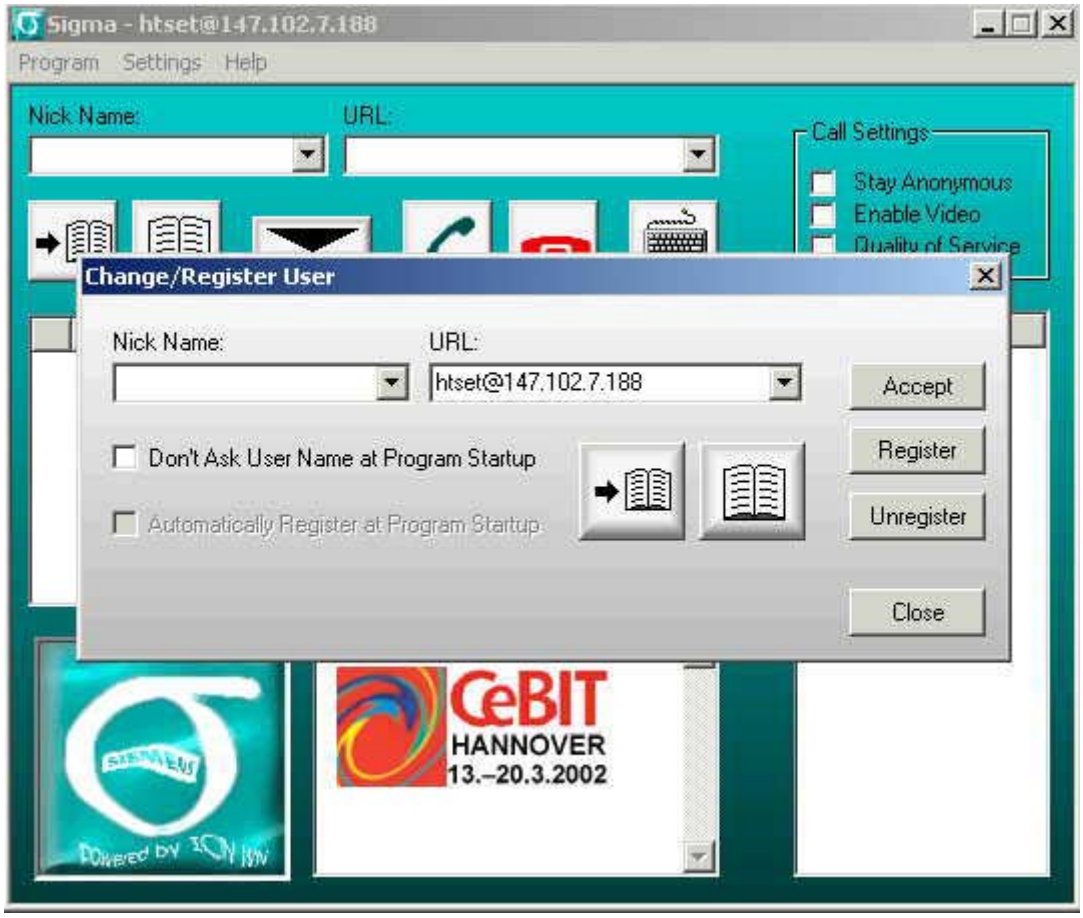

Figure 2-13: SIGMA initial dialog box

From the main window users can initiate SIP sessions, by entering the called party SIP address in the "URL:" field. This address should have the same format, as the one of the initiator: username@host-address. By pressing the arrow button, the message exchange through the SIP Proxy starts.

The progress of the call is displayed in the call status list in the middle of the window. If SIP signalling is successful and the called party's phone is ringing, then the symbol appears, instead of the ? in the call status list. If the call is accepted, then the symbol appears and the conversation may start.

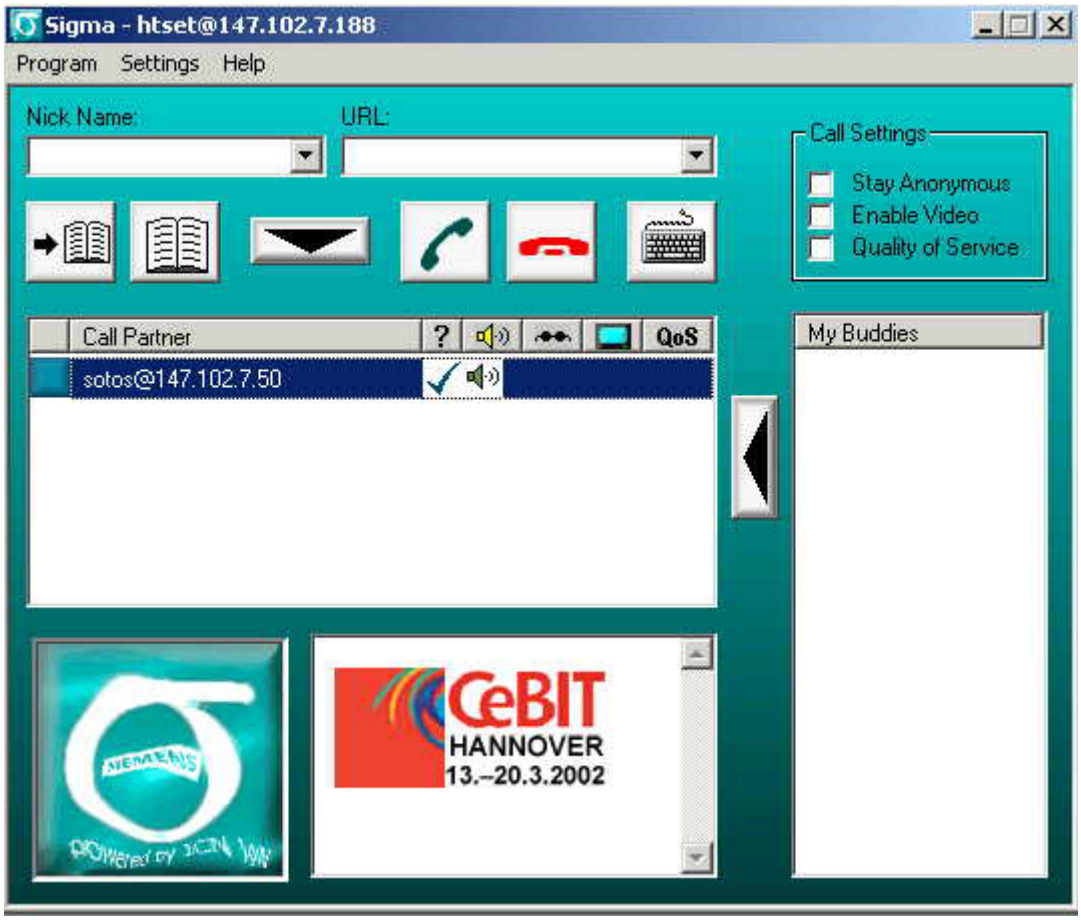

Figure 2-14: Successful call establishment

The SIP Proxy usage scenario comprises *QoS-enabled calls*. This means that if the reservation request towards the EAT is not accepted by the RCL (either for resources shortage or authentication failure), the SIP call continues normally without QoS. This is in contrast to the concept of *QoS-assured calls* where, if the request is denied, the SIP Proxy does not forward the SIP messages but generates error messages to the initiator of the call. The user may be informed of the reservation outcome by using the Graphical User Interfaces of the EAT.

3 Application Profiles

This chapter describes an important step towards Application Profiles that is, the analysis of the QoS requirements of the second trial applications[1], more specifically their service components. These requirements are put into tables mainly based on a common template which follow the ApplicationProfile.dtd and ServiceComponentProfile.dtd.

3.1 Mediazine

3.1.1 Profile for RealSystem

The Application Profile of the RealSystem includes two Service Component Profiles: one for the video component and one for the audio component. The video component offers the users three different qualities for the movies. These qualities are related to the bandwidth with which the video streams are coded. Mediazine offers the possibility to watch a movie in two screen formats:

1. a special format to dimension the movie into Part C of Mediazine by a resolution of the main browser window of 1024x768 pixels (see also chapter 2.1.2.1) and

2. the full screen format.

The bandwidth of the video stream does not change when a movie is presented in full screen mode. Therefore a option to change the QoS parameters in the Mediazine configuration menu for different video resolutions is not necessary and not implemented. The available options are:

OptionID 1: very low quality video stream is coded with 45 kBit/s

- This value is intended for a connection via 56 kBit/s modem, ISDN or dual ISDN. It corresponds to the bronze QoS level.

OptionID 2: medium quality video stream is coded with 384 kBit/s

- This value is intended for a connection via cable modem or DSL. It corresponds to the silver QoS level.

OptionID 3: high quality video stream is coded with 1024 kBit/s

- This value is intended for a connection via LAN. It corresponds to the gold QoS level.

[1] Here the Basic Internet Applications are meant. Although they may form together a Complex Internet Service (e.g. Mediazine) they act autonomously regarding IP connection and QoS.

The audio component offers also three different qualities for the pieces of music. The qualities are related to the bandwidth with which the audio streams are coded:

OptionID 1: very low quality audio stream is coded with 48 kBit/s

- This value is intended for a connection via 56 kBit/s modem or ISDN. It corresponds to the bronze QoS level.

OptionID 2: medium quality audio stream is coded with 112 kBit/s

- This value is intended for a connection via dual ISDN. It corresponds to the silver QoS level.

OptionID 3: high quality audio stream is coded with 256 kBit/s

- This value is intended for a connection via cable modem, DSL or LAN. It corresponds to the gold QoS level.

The entire Application Profile with all service components can be found in the following tables:

Application		
Name : RealSystem	**Version : 8.5**	**Build :**
Scope : unidirectional		
Type : MULTIMEDIA		
Comments:		
protocol : RTSP		
IsControlPort: 554	configurable	
Services components		
	name	TransportProtocol
1	RealSystem_8.5_Video_v01	UDP
2	RealSystem_8.5_Audio_v01	UDP

Table 3-1: RealSystem Application Profile

<table>
<tr><th colspan="6">ServiceComponent</th></tr>
<tr><td colspan="3">Name: RealSystem_8.5_Video_v01</td><td colspan="3">VIDEO</td></tr>
<tr><th colspan="6">QoSRequirement</th></tr>
<tr><td>(max) Delay</td><td>(max) Jitter</td><td>(max) Loss</td><td>packet ordering</td><td colspan="2">BW guarantee</td></tr>
<tr><td>in ms: 1000</td><td>in ms: 500</td><td>in percent: 5</td><td>false</td><td colspan="2">in percent: 90</td></tr>
<tr><td>veryLow</td><td>notRelevant</td><td>Medium</td><td></td><td colspan="2">high</td></tr>
<tr><td>2</td><td>0</td><td>5</td><td>5</td><td colspan="2">5</td></tr>
<tr><th colspan="6">Options</th></tr>
<tr><td colspan="6">optionID: 1
description: video low quality scenario
NetworkSpeed: 56kBit/s Modem / ISDN / Dual ISDN</td></tr>
<tr><td>SessionCharacteristic</td><td>TrafficSpecification</td><td colspan="4">AQUILASpecification</td></tr>
<tr><td>name: video quality</td><td>type: stream</td><td colspan="4">serviceID: PMM</td></tr>
<tr><td rowspan="3">semanticalGroup
type: UserFriendly
language: en
description: Video quality
qualifiers:
very low quality (45kBit/s)</td><td rowspan="3">duration: longLiving
adaptivity: true
burstiness: false</td><td colspan="4">BSP
in "bytes": 56000</td></tr>
<tr><td colspan="4">BSS
in "bytes": 5625</td></tr>
<tr><td colspan="4">minPU
in "bytes": 1</td></tr>
<tr><td rowspan="4">semanticalGroup
type: UserFriendly
language: de
description: Video-Qualität
qualifiers:
sehr niedrige Qualität (45kBit/s)</td><td>packetSize
variability: variable
qualitatively: small
• averagePacketSize
in "bytes: medium
• maximumPacketSize
in "bytes": veryBig
• minimumPolicedUnit
in "bytes": medium</td><td colspan="4">maxPS
in "bytes": 1500</td></tr>
<tr><td colspan="4">PR
in "bit/s": 56000</td></tr>
<tr><td colspan="4">SR
in "bit/s": 46080</td></tr>
<tr><td>bitRate
variability: variable
qualitatively: low
• peakRate
in "bit/s": low
• averageRate
in "bit/s": low</td><td colspan="4"></td></tr>
<tr><td colspan="2">flow: non-greedy</td><td colspan="4"></td></tr>
<tr><td colspan="6">optionID: 2
description: video medium quality scenario
NetworkSpeed: Cable Modem / DSL</td></tr>
<tr><td>SessionCharacteristic</td><td>TrafficSpecification</td><td colspan="4">AQUILASpecification</td></tr>
<tr><td>name: video quality</td><td>type: stream</td><td colspan="4">serviceID: PMM</td></tr>
<tr><td rowspan="3">semanticalGroup
type: UserFriendly
language: en
description: Video quality
qualifiers:
medium quality (384kBit/s)</td><td rowspan="3">duration: longLiving
adaptivity: true
burstiness: false</td><td colspan="4">BSP
in "bytes": 480000</td></tr>
<tr><td colspan="4">BSS
in "bytes": 49152</td></tr>
<tr><td colspan="4">minPU
in "bytes": 7</td></tr>
<tr><td rowspan="3">semanticalGroup
type: UserFriendly
language: de
description: Video-Qualität
qualifiers:
mittlere Qualität (384kBit/s)</td><td rowspan="3">packetSize
variability: variable
qualitatively: medium
• averagePacketSize
in "bytes: medium</td><td colspan="4">maxPS
in "bytes": 1500</td></tr>
<tr><td colspan="4">PR
in "bit/s": 480000</td></tr>
<tr><td colspan="4">SR
in "bit/s": 393216</td></tr>
</table>

	• maximumPacketSize in "bytes": veryBig • minimumPolicedUnit in "bytes": medium	
	bitRate variability: variable qualitatively: medium • peakRate in "bit/s": low • averageRate in "bit/s": low	
	flow: non-greedy	
optionID: 3 **description: video high quality scenario** **NetworkSpeed: LAN**		
SessionCharacteristic	**TrafficSpecification**	**AQUILASpecification**
name: video quality	type: stream	serviceID: PMM
semanticalGroup type: UserFriendly language: en description: Video quality qualifiers: high quality (1024kBit/s)	duration: longLiving adaptivity: true burstiness: false	BSP in "bytes": 1280000
		BSS in "bytes": 131072
		minPU in "bytes": 19
semanticalGroup type: UserFriendly language: de description: Video-Qualität qualifiers: hohe Qualität (1024kBit/s)	**packetSize** variability: variable qualitatively: big • averagePacketSize in "bytes: big • maximumPacketSize in "bytes": veryBig • minimumPolicedUnit in "bytes": medium	maxPS in "bytes": 1500
		PR in "bit/s": 1280000
		SR in "bit/s": 1048576
	bitRate variability: variable qualitatively: high • peakRate in “bit/s”: low • averageRate in “bit/s”: low	
	flow: non-greedy	

Table 3-2: RealSystem Service Component Profile VIDEO

<table>
<tr><th colspan="6">ServiceComponent</th></tr>
<tr><td colspan="3">Name: RealSystem_8.5_Audio_v01</td><td colspan="3">AUDIO</td></tr>
<tr><th colspan="6">QoSRequirement</th></tr>
<tr><td>(max) Delay</td><td>(max) Jitter</td><td>(max) Loss</td><td>packet ordering</td><td colspan="2">BW guarantee</td></tr>
<tr><td>in ms: 1000</td><td>in ms: 500</td><td>in percent: 5</td><td>false</td><td colspan="2">in percent: 90</td></tr>
<tr><td>veryLow</td><td>notRelevant</td><td>medium</td><td></td><td colspan="2">high</td></tr>
<tr><td>2</td><td>0</td><td>5</td><td>5</td><td colspan="2">5</td></tr>
<tr><th colspan="6">Options</th></tr>
<tr><td colspan="6">optionID: 1
description: audio low quality scenario
NetworkSpeed: 56kBit/s Modem / ISDN</td></tr>
<tr><td>SessionCharacteristic</td><td colspan="2">TrafficSpecification</td><td colspan="3">AQUILASpecification</td></tr>
<tr><td>name: audio quality</td><td colspan="2">type: stream</td><td colspan="3">serviceID: PMM</td></tr>
<tr><td rowspan="3">semanticalGroup
type: UserFriendly
language: en
description: Audio quality
qualifiers:
very low quality (48kBit/s)</td><td colspan="2" rowspan="3">duration: longLiving
adaptivity: true
burstiness: false</td><td colspan="3">BSP
in “bytes”: 60000</td></tr>
<tr><td colspan="3">BSS
in “bytes”: 6144</td></tr>
<tr><td colspan="3">minPU
in “bytes”: 1</td></tr>
<tr><td rowspan="3">semanticalGroup
type: UserFriendly
language: de
description: Audio-Qualität
qualifiers:
sehr niedrige Qualität (48kBit/s)</td><td colspan="2">packetSize
variability: constant
qualitatively: small
• averagePacketSize
in "bytes: medium
• maximumPacketSize
in "bytes": veryBig
• minimumPolicedUnit
in "bytes": medium</td><td colspan="3">maxPS
in "bytes": 1500</td></tr>
<tr><td colspan="3">PR
in "bit/s": 60000</td></tr>
<tr><td colspan="3">SR
in "bit/s": 49152</td></tr>
<tr><td></td><td colspan="2">bitRate
variability: constant
qualitatively: low
• peakRate
in "bit/s": low
• averageRate
in "bit/s": low</td><td colspan="3"></td></tr>
<tr><td></td><td colspan="2">flow: non-greedy</td><td colspan="3"></td></tr>
<tr><td colspan="6">optionID: 2
description: audio medium quality scenario
NetworkSpeed: Dual ISDN</td></tr>
<tr><td>SessionCharacteristic</td><td colspan="2">TrafficSpecification</td><td colspan="3">AQUILASpecification</td></tr>
<tr><td>name: audio quality</td><td colspan="2">type: stream</td><td colspan="3">serviceID: PMM</td></tr>
<tr><td rowspan="3">semanticalGroup
type: UserFriendly
language: en
description: Audio quality
qualifiers:
medium quality (112kBit/s)</td><td colspan="2" rowspan="3">duration: longLiving
adaptivity: true
burstiness: false</td><td colspan="3">BSP
in “bytes”: 140000</td></tr>
<tr><td colspan="3">BSS
in “bytes”: 14336</td></tr>
<tr><td colspan="3">minPU
in “bytes”: 2</td></tr>
<tr><td rowspan="3">semanticalGroup
type: UserFriendly
language: de
description: Audio-Qualität
qualifiers:
mittlere Qualität (112kBit/s)</td><td colspan="2" rowspan="3">packetSize
variability: constant
qualitatively: small
• averagePacketSize
in "bytes: medium</td><td colspan="3">maxPS
in "bytes": 1500</td></tr>
<tr><td colspan="3">PR
in "bit/s": 140000</td></tr>
<tr><td colspan="3">SR
in "bit/s": 114688</td></tr>
</table>

	• maximumPacketSize in "bytes": veryBig • minimumPolicedUnit in "bytes": medium	
	bitRate variability: constant qualitatively: low • peakRate in "bit/s": low • averageRate in "bit/s": low	
	flow: non-greedy	
optionID: 3 **description: audio high quality scenario** **NetworkSpeed: Cable Modem / DSL / LAN**		
SessionCharacteristic	**TrafficSpecification**	**AQUILASpecification**
name: audio quality	type: stream	serviceID: PMM
semanticalGroup type: UserFriendly language: en description: Audio quality qualifiers: high quality (256kBit/s)	duration: longLiving adaptivity: true burstiness: false	BSP in “bytes”: 320000
		BSS in “bytes”: 32768
		minPU in “bytes”: 5
semanticalGroup type: UserFriendly language: de description: Audio-Qualität qualifiers: hohe Qualität (256kBit/s)	**packetSize** variability: constant qualitatively: medium • averagePacketSize in "bytes: big • maximumPacketSize in "bytes": veryBig • minimumPolicedUnit in "bytes": medium	maxPS in "bytes": 1500
		PR in "bit/s": 320000
		SR in "bit/s": 262144
	bitRate variability: constant qualitatively: medium • peakRate in "bit/s": low • averageRate in "bit/s": low	
	flow: non-greedy	

Table 3-3: RealSystem Service Component Profile AUDIO

3.1.2 Profile for NetMeeting

The NetMeeting component is integrated into Mediazine to test the AQUILA architecture by using two basic Internet applications which needs different QoS configurations at the same time. Mediazine supports only a video conference connection between two users at the same time via video mode with a default resolution for the video screens. It is possible to select other modes (the audio mode) as well as other video screen resolutions by the NetMeeting set-up menu. These other modes and resolutions are not checked and not supported by Medi-

azine. Therefore the Application Profile of the NetMeeting includes only one Service Component Profile for a video mode connection with one default video screen size The connection is supported by three different options. These options are:

OptionID 1: very low quality

- This value is intended for a connection via 56 kBit/s modem and the NetMeeting set-up for a 14.4 kBit/s modem and for a 28.8 kBit/s modem. It corresponds to the bronze QoS level.

OptionID 2: medium quality

- This value is intended for a connection via ISDN or dual ISDN and the NetMeeting set-up for ISDN. It corresponds to the silver QoS level.

OptionID 3: high quality the video stream is coded with 1024 kBit/s

- This value is intended for a connection via cable modem, DSL or LAN and the NetMeeting set-up for LAN. It corresponds to the gold QoS level.

The entire Application Profile with all service components can be found in the following tables:

Application		
Name : NetMeeting	**Version : 3.01**	**Build : 4.4.3388**
Scope : xdirectional		
Type : MULTIMEDIA		
Comments:		
protocol : H323		
IsControlPort: 1720	fixed	
protocol : RTP		
IsControlPort:	fixed	
Services components		
	Name	TransportProtocol
1	NetMeeting_3.01_Video_v01	TCP

Table 3-4: NetMeeting Application Profile

<table>
<tr><th colspan="6">ServiceComponent</th></tr>
<tr><td colspan="3">Name: NetMeeting_3.01_Video_v01</td><td colspan="3">VIDEO</td></tr>
<tr><th colspan="6">QoSRequirement</th></tr>
<tr><td>(max) Delay</td><td>(max) Jitter</td><td>(max) Loss</td><td>packet ordering</td><td colspan="2">BW guarantee</td></tr>
<tr><td>in ms: 1200</td><td>in ms: 120</td><td>in percent: 10</td><td>true</td><td colspan="2">in percent: 0</td></tr>
<tr><td>high</td><td>low</td><td>medium</td><td></td><td colspan="2">high</td></tr>
<tr><td>1</td><td>3</td><td>5</td><td>8</td><td colspan="2">8</td></tr>
<tr><th colspan="6">Options</th></tr>
<tr><td colspan="6">optionID: 1
description: video low quality scenario
NetworkSpeed: 56kBit/s Modem</td></tr>
<tr><td>SessionCharacteristic</td><td colspan="2">TrafficSpecification</td><td colspan="3">AQUILASpecification</td></tr>
<tr><td>name: video quality</td><td colspan="2">type: elastic</td><td colspan="3">serviceID: PVBR</td></tr>
<tr><td rowspan="3">semanticalGroup
type: UserFriendly
language: en
description: Video quality
qualifiers:
very low quality (28.8kBit/s)</td><td colspan="2" rowspan="3">duration: longLiving
adaptivity: false
burstiness: true</td><td colspan="3">BSP
in “bytes”: 2000</td></tr>
<tr><td colspan="3">BSS
in “bytes”: 2048</td></tr>
<tr><td colspan="3">minPU
in “bytes”: 60</td></tr>
<tr><td rowspan="3">sehr niedrige Qualität (28.8kBit/s)</td><td colspan="2">packetSize
variability: variable
qualitatively: medium
• averagePacketSize
in "bytes: medium
• maximumPacketSize
in "bytes": medium
• minimumPolicedUnit
in "bytes": small</td><td colspan="3">maxPS
in "bytes": 1500
PR
in "bit/s": 28800
SR
in "bit/s": 19200</td></tr>
<tr><td colspan="2">bitRate
variability: variable
qualitatively: high
• peakRate
in "bit/s": medium
• averageRate
in "bit/s": medium</td><td colspan="3"></td></tr>
<tr><td colspan="2">flow: greedy</td><td colspan="3"></td></tr>
<tr><td colspan="6">optionID: 2
description: video medium quality scenario
NetworkSpeed: ISDN / Dual ISDN</td></tr>
<tr><td>SessionCharacteristic</td><td colspan="2">TrafficSpecification</td><td colspan="3">AQUILASpecification</td></tr>
<tr><td>name: video quality</td><td colspan="2">type: elastic</td><td colspan="3">serviceID: PVBR</td></tr>
<tr><td rowspan="3">semanticalGroup
type: UserFriendly
language: en
description: Video quality
qualifiers:
medium quality (64kBit/s)</td><td colspan="2" rowspan="3">duration: longLiving
adaptivity: false
burstiness: true</td><td colspan="3">BSP
in “bytes”: 2000</td></tr>
<tr><td colspan="3">BSS
in “bytes”: 4096</td></tr>
<tr><td colspan="3">minPU
in “bytes”: 60</td></tr>
<tr><td rowspan="3">mittlere Qualität (64kBit/s)</td><td colspan="2">packetSize
variability: variable
qualitatively: medium
• averagePacketSize
in "bytes: medium</td><td colspan="3">maxPS
in "bytes": 1500
PR
in "bit/s": 64000
SR
in "bit/s": 48000</td></tr>
<tr><td colspan="2"></td><td colspan="3"></td></tr>
<tr><td colspan="2"></td><td colspan="3"></td></tr>
</table>

	• maximumPacketSize in "bytes": big • minimumPolicedUnit in "bytes": medium	
	bitRate variability: variable qualitatively: high • peakRate in "bit/s": high • averageRate in "bit/s": medium	
	flow: greedy	
optionID: 3 **description: video high quality scenario** **NetworkSpeed: Cable Modem / DSL / LAN**		
SessionCharacteristic	**TrafficSpecification**	**AQUILASpecification**
name: video quality	type: elastic	serviceID: PVBR
semanticalGroup type: UserFriendly language: en description: Video quality qualifiers: high quality (160kBit/s)	duration: longLiving adaptivity: false burstiness: true	BSP in “bytes”: 2000
		BSS in “bytes”: 5120
		minPU in “bytes”: 60
semanticalGroup type: UserFriendly language: de description: Video-Qualität qualifiers: hohe Qualität (160kBit/s)	**packetSize** variability: variable qualitatively: big • averagePacketSize in "bytes: big • maximumPacketSize in "bytes": veryBig • minimumPolicedUnit in "bytes": medium	maxPS in "bytes": 1500
		PR in "bit/s": 160000
		SR in "bit/s": 75000
	bitRate variability: variable qualitatively: high • peakRate in "bit/s": high • averageRate in "bit/s": high	
	flow: greedy	

Table 3-5: NetMeeting Service Component Profile VIDEO

3.1.3 Profile for the game OIDS

The requirements of the OIDS game are very low. They can be full filled by guaranteeing a connection with 28.8 kBit/s or more. Therefore the Application Profile of the OIDS game includes only one Service Component Profile for enabling the QoS support. The available option is:

OptionID 1: QoS

- This value is intended to enable the QoS support for all kind of connections. It corresponds to all QoS levels.

The entire Application Profile with all service components can be found in the following tables:

Application		
Name : OIDS	**Version : unknown**	**Build :**
Scope : xdirectional		
Type : GAME		
Comments:		
Services components		
	Name	TransportProtocol
1	OIDS-Game_Data_v01	TCP

Table 3-6: OIDS Game Application Profile

<table>
<tr><td colspan="6">ServiceComponent</td></tr>
<tr><td colspan="3">Name: OIDS-Game_Data_v01</td><td colspan="3">DATA</td></tr>
<tr><td colspan="6">QoSRequirement</td></tr>
<tr><td>(max) Delay</td><td>(max) Jitter</td><td>(max) Loss</td><td>packet ordering</td><td colspan="2">BW guarantee</td></tr>
<tr><td>in ms: 100</td><td>in ms: 20</td><td>in percent: 2</td><td>true</td><td colspan="2">in percent: 90</td></tr>
<tr><td>low</td><td>low</td><td>low</td><td></td><td colspan="2">high</td></tr>
<tr><td>8</td><td>5</td><td>8</td><td>7</td><td colspan="2">5</td></tr>
<tr><td colspan="6">Options</td></tr>
<tr><td colspan="6">optionID: 1
description: QoS enabled
NetworkSpeed: 56kBit/s Modem / ISDN / Dual ISDN / Cable Modem / DSL /LAN</td></tr>
<tr><td>SessionCharacteristic</td><td colspan="2">TrafficSpecification</td><td colspan="3">AQUILASpecification</td></tr>
<tr><td>name: QoS</td><td colspan="2">type: realTime</td><td colspan="3">serviceID: PMC</td></tr>
<tr><td rowspan="3">semanticalGroup
type: UserFriendly
language: en
description: QoS
qualifiers:
enabled</td><td colspan="2" rowspan="3">duration: shortLiving
adaptivity: false
burstiness: false</td><td colspan="3">BSP
in “bytes”: 40000</td></tr>
<tr><td colspan="3">BSS
in “bytes”: 2048</td></tr>
<tr><td colspan="3">minPU
in “bytes”: 1</td></tr>
<tr><td rowspan="3">semanticalGroup
type: UserFriendly
language: de
description: QoS
qualifiers:
eingeschaltet</td><td colspan="2">packetSize
variability: constant
qualitatively: small
• averagePacketSize
in "bytes: small
• maximumPacketSize
in "bytes": medium
• minimumPolicedUnit
in "bytes": medium</td><td colspan="3">maxPS
in "bytes": 1500</td></tr>
<tr><td colspan="2">bitRate
variability: constant
qualitatively: low
• peakRate
in "bit/s": medium
• averageRate
in "bit/s": low</td><td colspan="3">PR
in "bit/s": 40000
SR
in "bit/s": 2048</td></tr>
<tr><td colspan="2">flow: non-greedy</td><td colspan="3"></td></tr>
</table>

Table 3-7: OIDS Game Service Component Profile DATA

3.2 SIGMA

3.2.1 Codecs

At present, the automatic support of SIP applications such as SIGMA is the task of the EAT’s Converter component. This support is hidden to the end-user, i.e. the end-user is not involved in choosing a QoS level for its session, since the most available SIP applications are not QoS-aware. Consequently, there is no strong need for Application Profiles, because it is not necessary to map between the end-user’s QoS level and the technical AQUILA level.

Instead, simple codec profiles are used within the converter to support the different codecs that can be used by SIP applications.

The following table depicts these profiles that were estimated for the codecs used by the SIP application SIGMA:

	NS	**PR**	**BSP**
PCMA/8000	PCBR	85000	1600
PCMU/8000	PCBR	85000	1600
GSM/8000	PCBR	40000	3250
H263/90000	PVBR	320000	18130
MPEG4/90000	PVBR	320000	18130

Table 3-8: Codec profiles for SIGMA

The values of SR (Sustainable Rate) and BSS (Bucket Size for Sustainable rate) are -1 for all codecs.

The above depicted codec profiles can be the base for new, reusable Service Component Profiles later on, for example one Service Component Profile for each codec. Application Profiles for upcoming SIP applications that are QoS-aware can then refer to a sub-set of these profiles or specify a new/modified one depending on which codec(s) the application implements.

4 Summary

This deliverable presented the user applications for the second AQUILA trial. The innovation is twofold: On the one hand, a new CIS Mediazine uses and implements the Application Profile approach to present the QoS at end-user level. On the other hand, the SIP proxy reserves for QoS on behalf of the application and the end-user for a SIP application.

The generic part of the Application Profile especially the part aimed for the metaphors' description is easy to use and adequate for the CIS developers. Nevertheless the reverse engineering of legacy applications in order to determine the traffic specification and the implementation specific AQUILA specification is not an easy task. With the Application Profile approach it is possible to supply non-QoS-aware applications with QoS.

The CIS Mediazine is just an example on how the usage of the Application Profile approach together with the EAT can easily bring QoS to Internet applications and services. Due to the fact that the profiles are not part of the CIS but independently provided by the EAT for several applications, the reuse of this approach is quite simple to integrate: There is no need to modify the core of existing CIS, instead, the QoS support can be implemented at user interface level as shown in Mediazine. Also the integration of the EAT API (necessary to retrieve the profiles and to send reservation requests) consist of a few lines of code only.

The next step for the Application Profiles consists in populating a repository of Application Profiles for legacy applications. Due to the fact that the profiles of Mediazine are for very popular Basic Internet Applications such as RealSystem (RealPlayer/Server) and NetMeeting, the already existing profiles are an important outcome of the AQUILA project. Additional Application Profiles and Service Component Profiles (for example based on codecs) may follow.

However, profiles which are based on codecs can only be *templates* for "usable" Service Component Profiles. The most Internet applications have in fact very specific QoS requirements, even if they use common codecs. To be more concrete, there will probably a need for specific Application Profiles as well as Service Component Profiles for *each* existing Basic Internet Application that is foreseen to be provided with QoS. However, existing profile templates can enforce the creation process of specific profiles, since they can easily be modified.

A further step goes towards development of QoS-aware applications using the Application Profile approach. This implies that the Application Profile concept is absolutely generic in order to enable a mapping towards QoS enabled networks.

Regarding the SIP support, the benefit of the Proxy approach is that several existing SIP applications can be provided with QoS in this way. SIGMA is just an example on how the automatic support works. Moreover, the Proxy approach works fine not only for SIP but also for other signalling protocols, as the H.323 Proxy for NetMeeting proved in the first trial. Additional protocol gateways (here also called (application) proxies) for other protocols can be integrated by using the existing Proxy Framework.

5 Abbreviations

A

API	Application Programming Interface
AQUILA	Adaptive Resource Control for QoS Using an IP-based Layered Architecture

B

BIA	Basic Internet Application (Legacy Application)
BSP	Bucket Size for PR
BSS	Bucket Size for SR
BW	Bandwidth

C

CIS	Complex Internet Service
Codec	COmpression/DECompression

D

DSL	Digital Subscriber Line
DTD	Document Type Definition

E

EAT	End-user Application Toolkit

G

GUI	Graphical User Interface

I

IETF	Internet Engineering Task Force
IMP	Internet Messaging Program
IMAP	Internet Message/Mail Access Protocol
IP	Internet Protocol
ISDN	Integrated Services Digital Network

L

LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol

M

maxPS	Maximum Packet Size
minPU	minimum Policed Unit

N

NS	Network Service

P

PCBR	Premium CBR
PMC	Premium Mission Critical
PMM	Premium MultiMedia
PR	Peak Rate
PVBR	Premium VBR

Q

QoS	Quality of Service

R

RCL	Resource Control Layer
RTP	Real Time Protocol
RTSP	Real Time Streaming Protocol

S

SIGMA	SIP-based IntelliGent Multimedia Application
SIP	Session Initiation Protocol
SIP UA	SIP User Agent
SQL	Structured Query Language
SR	Sustainable Rate

T

TCP	Transmission Control Protocol

U

UDP	User Datagram Protocol
URL	Uniform Resource Locator

V

VoIP	Voice over IP

X

XML	eXtensible Markup Language

6 References

[D1203] IST-1999-10077-WP1.2-SAG-1203-PU-O/b0, *"Final system specification"*, M. Winter (ed.), April 2002

[D2202] IST-1999-10077-WP2.2-TUD-2202-PU-O/b0, *"Description of user applications for the first trial",* F. Fünfstück (ed.), March 2001

[D2203] IST-1999-10077-WP2.2-TUD-2203-PU-R/b0, *"User Guide for End-user Application Toolkit",* F. Fünfstück (ed.), December 2001

[Manyfolks] G. Camarillo (ed.), W. Marshall (ed.), Jonathan Rosenberg, "*Integration of Resource Management and SIP*", Internet draft, draft-ietf-sip-manyfolks-resource-07.txt, April 2002

[RFC2543] M. Handley et al., "*SIP: Session Initiation Protocol*", http://www.ietf.org/rfc/rfc2543.txt

Appendix

New issues on the usage of the Application Interfaces

EAT API

The EAT API has a few but important modifications to the version presented in [D2203]:

- A `QoSOption` is now a structure containing a unique pair of an option id and the (unique) name of the belonging service component which refers to a Service Component Profile. The method `prepareSessionCharactericticsOptions()` of the interface `QoSSessionRequest` returns such a `QoSOption`. It is also used within the different methods for regular reservation requests.

- The interface `ApplicationManager` additionally provides three methods to retrieve the Application Profiles from the EAT: `getAvailableSCs()` is to get meta information about the available Service Component Profiles of one Application Profile (incl. all options; and similar to `getAvailableApps()`), `getApplicationProfile()` is to get the reference to the Application Profile object in order to have direct access to all of its data; and `getProfileStream()` is to get the entire profile as byte stream, in order to parse or unmarshal it for example.

- The new structure `NetworkServiceInformation` (that replaces `NetworkService`) contains the most important information of a network service. Such information for all network services can be retrieved by using the method `getNetworkServices()` of the interface `ServiceDistributor`.

- The interface `ServiceDistributor` additionally provides a method to retrieve the network services from the EAT: `getNetworkService()` returns a reference to the network service object in order to have direct access to all of its data.

```
/*
    Package aquila.rcl.eat.api
    File    api.idl
    Version 4.3, 31.05.2002

    Dresden University of Technology (TUD),
    National Technical University of Athens (NTUA)

    This file contains the specification of the (internal) api
    of the End-user Application Toolkit (EAT) for the 2nd trial.

    It is intended for the use by the applications as well as the
    Reservation GUI, etc.
*/

#ifndef   api_idl
#define   api_idl
```

```
#include "service.idl"
#include "subscriber.idl"
#include "aca.idl"
#include "appProfile.idl"
#include "converter.idl"
#include "eatPersistence.idl"
#include "proxy.idl"
#include "event.idl"

module api {

    /* --- Forward declarations --- */

    interface QoSSessionRequest;
    interface QoSSession;
    interface QoSSessionGroup;
    interface QoSSessionUnit;

    typedef sequence<QoSSession> QoSSessionSeq;
    typedef sequence<QoSSessionUnit> QoSSessionUnitSeq;

    /**
     * The QoSRequester interface should be implemented by the QoS requesting
     * client in or der to have the chance to inform it about some events
     * concerning the requested reservation, e.g. when a provisional reservation
     * has been accepted/rejected by the ACA, or when a reservation brokes, etc.
     */
    typedef event::EventObserver QoSRequester;

    /**
     * APIException containing a message string.
     */
    exception APIException {
        string reason;
    };


    /* --- Common --- */

    typedef service::ServiceIDSeq ServiceIDSeq;
    typedef sequence<aca::SLS>    slsSeq;
    typedef sequence<aca::Scope>  ScopeSeq;
    typedef sequence<aca::Flow>   FlowSeq;
    typedef sequence<octet> XMLStream;


    /* --- Login ------------------------------------------------------------ */

    /**
     * The Login interface allows the authentication of an end-user against
     * the EAT (Manager). It is implemented by the EAT.
     * The login information is also forwarded to the ACA.
     */
    interface Login {

        const string SERVER_PREFIX = "rcl.directory/eat.directory/api.directory/";
        const string SERVER_SUFFIX = ".Login";

        /**
         * Login at the EAT Manager's API.
         *
         * @param   loginInfo        Account name and password.
         * @param   clientSessionId  Created by a Web server providing the
         *                           Login GUI.
         * @return                   Reference to a new QoSSessionRequest
         *                           object.
         */
```

```
        QoSSessionRequest loginClient (
            in subscriber::LoginInfo logInfo,
            in string                clientSessionId)
            raises (APIException);

        /**
         * Gets the reference to the QoSSessionRequest object created at the
         * login.
         * Called by the Reservation GUI, the Regular Reservation GUI, etc.
         *
         * @param   clientSessionId  Web servers' session id, included in the
         *                           URL.
         * @return                   Reference to the existing QoSSessionRequest
         *                           object.
         */
        QoSSessionRequest getQSRequestMeansSessionId (
            in string clientSessionId);
    };


    /* --- QoS Session Request ------------------------------------------- */

    typedef sequence<string> ClientSessionIdSeq;

    /**
     * The QoSSessionRequest interface is the user agent for QoS session
     * requests and SLA retrieval.
     * Requests can be made on an advanced and on a regular level.
     */
    interface QoSSessionRequest {

        /**
         * The subscriber.
         */
        readonly attribute string accountName;

        /**
         * All current client session ids.
         */
        readonly attribute ClientSessionIdSeq clientSessionIds;


        /* --- SLAs --- */

        /**
         * The SLAs.
         */
        readonly attribute converter::SLASeq contracts;


        /* --- Advanced Request --- */

        /**
         * For the specification of a (single) advanced request.
         *
         * applicationId     Name that the end-user uses to identify the
         *                   application.
         * serviceComponent  Service component that this reservation corresponds
         *                   to.
         * networkService    Network service id.
         * reqSLS            Requested SLS incl, scope, flow, traffic spec, QoS
         *                   spec, and schedule.
         * proxyId           Id of the proxy to be used (see ApplicationProxy),
         *                   0 for none.
         * enabled           Indicates whether the reservation shall be
         *                   immediately established in the edge router or not.
         */
```

```
struct AdvancedSpec {
    string             applicationId;     // One application
    string             serviceComponent;  // One service component
    service::ServiceID networkService;    // One service
    aca::SLS           reqSLS;            // One SLSpec
    long               proxyId;           // One proxy
    boolean            enabled;
};

/**
 * Requests for a (single) QoSSession on advanced level.
 * It is foreseen for reservation requests that are based
 * on the content of the ACA's reservation request interface,
 * the so-called Advanced Reservation Mode.
 *
 * @param  requestSpec  The requested parameters.
 * @param  requester    The requester object that has to be notified
 *                      when something happens with the reservation.
 *                      Can be null.
 * @return              Reference to a new QoSSessionUnit object with
 *                      one single element.
 */
QoSSessionUnit advancedRequest (
    in AdvancedSpec requestSpec,
    in QoSRequester requester)
    raises (APIException);


/* --- Advanced Multiple Request --- */

/**
 * For the specification of a multiple (e.g. bi-directional)
 * advanced request.
 *
 * applicationId     Name that the end-user uses to identify the
 *                   application.
 * serviceComponent  Service component that this reservation corresponds
 *                   to.
 * networkServices   Network service ids.
 * reqSLSs           Requested SLSs incl. scoped, flows, traffic specs,
 *                   QoS specs, and schedules.
 * proxyId           Id of the proxy to be used (see ApplicationProxy),
 *                   0 for none.
 * enabled           Indicates whether the reservation shall be
 *                   immediately established in the edge router or not.
 */
struct AdvancedMultipleSpec {
    string             applicationId;     // One application
    string             serviceComponent;  // One service component
    ServiceIDSeq       networkServices;   // Several services !
    slsSeq             reqSLSs;           // Several SLSpecs !
    long               proxyId;           // One proxy
    boolean            enabled;
};

/**
 * Advanced request for a multiple (e.g. a bi-directional) reservation,
 * building an inseperable unit of session elements, the session unit.
 *
 * @param  requestSpec  The requested parameters.
 * @param  requester    The requester object that has to be notified
 *                      when something happens with the reservation.
 *                      Can be null.
 * @return              Reference to a new QoSSessionUnit object with
 *                      as many elements as requested. In the case of a
 *                      bi-directional session, there are two request
 *                      elements.
```

```
         */
        QoSSessionUnit advancedMultipleRequest (
            in AdvancedMultipleSpec requestSpec,
            in QoSRequester         requester)
            raises (APIException);


        /* --- Advanced Group Request --- */

        /**
         * For the specification of a group request.
         */
        typedef sequence<AdvancedMultipleSpec> AdvancedSpecSeq;

        /**
         * Advanced request for a group of QoS sessions.
         *
         * @param  groupName     The optional name of the new group (can be "").
         * @param  requestSpecs  An array of (multiple) request specifications.
         * @param  requester     The requester object that has to be notified
         *                       when something happens with the reservation.
         *                       Can be null.
         * @return               An new QoSSessionGroup object, containing
         *                       one QoSSessionUnit object for each (multiple)
         *                       request.
         */
        QoSSessionGroup advancedGroupRequest (
            in string          groupName,
            in AdvancedSpecSeq requestSpecs,
            in QoSRequester    requester)
            raises (APIException);


        /* --- Regular Request --- */

        /**
         * Prepares the Regular App GUI with the options from the profile,
         * gets the ids for the session characteristics options to be displayed.
         *
         * @param  applicationProfile  ID of the associated Application
         *                             Profile.
         * @return                     An array of options of the profiles.
         */
        converter::QoSOptionSeq prepareSessionCharacteristicsOptions (
            in appProfile::ProfileID applicationProfile)
            raises (APIException);

        /**
         * For the specification of a (single) regular request.
         * Note that for each service component one request is needed.
         *
         * applicationProfile  ID of the associated Application Profile.
         * applicationId       The application name within the application
         *                     Profile.
         * selection           Selected session characteristic option from the
         *                     profile incl. the service component.
         * reqScope            Scope: reservation style.
         * reqFlow             Source and dest. addresses, ports, protocol,
         *                     DSCP.
         * schedule            Service schedule: reservation time.
         * proxyName           Name of the proxy to be used (see Application
         *                     Profile), the EAT Manager can look into the *.rc
         *                     file in order to get the id of the proxy;
         *                     "" if no proxy is needed.
         * enabled             Indicates whether the reservation shall be
         *                     immediately established in the edge router or
         *                     not.
```

```
 */
struct RegularSpec {
    appProfile::ProfileID applicationProfile;  // One
    string                applicationId;       //  application
    converter::QoSOption  selection;           // One service component
    aca::Scope            reqScope;            // One scope
    aca::Flow             reqFlow;             // One flow spec
    aca::ServiceSchedule  schedule;            // One schedule
    string                proxyName;           // One proxy
    boolean               enabled;
};

/**
 * Requests for a QoSSession on regular level.
 * It is foreseen for applications which are not QoS-aware
 * but are supported by Application Profiles for manual,
 * non-professional reservations. The end-user has to ask for the
 * preparation of suitable Session Charactersitics options and requests
 * then for such as QoS session.
 *
 * @param  requestSpec  The requested parameters.
 * @param  requester    The requester object that has to be notified
 *                      when something happens with the reservation.
 *                      Can be null.
 * @return              Reference to a new QoSSessionUnit object
 *                      containing one element for the reservation of
 *                      the specified service component. (For several
 *                      service components, use regularGroupRequest().)
 */
QoSSessionUnit regularRequest (
    in RegularSpec    requestSpec,
    in QoSRequester requester)
    raises (APIException);


/* --- Regular Multiple Request --- */

/**
 * For the specification of a multiple (e.g. bi-directional) regular
 * request.
 *
 * applicationProfile  ID of the associated Application Profile.
 * applicationId       The application name within the Application
 *                     Profile.
 * selection           Selected session characteristic option from the
 *                     profile incl. the service component.
 * reqScopes           Scopes: reservation styles.
 * reqFlows            Source and dest. addresses, ports, protocol,
 *                     DSCP.
 * schedule            Service schedule: reservation time.
 * proxyName           Name of the proxy to be used (see Application
 *                     Profile), the EAT Manager can look into the *.rc
 *                     file in order to get the id of the proxy;
 *                     "" if no proxy is needed.
 * enabled             Indicates whether the reservation shall be
 *                     immediately established in the edge router or
 *                     not.
 */
struct RegularMultipleSpec {
    appProfile::ProfileID applicationProfile;  // One
    string                applicationId;       //  application
    converter::QoSOption  selection;           // One service component
    ScopeSeq              reqScopes;           // Several scopes !
    FlowSeq               reqFlows;            // Several flow specs !
    aca::ServiceSchedule  schedule;            // One schedule
    string                proxyName;           // One proxy
    boolean               enabled;
```

```
};

/**
 * Regular request for a multiple (e.g. a bi-directional) reservation,
 * building an inseperable unit of session elements, the session unit.
 *
 * @param  requestSpec  The requested parameters.
 * @param  requester    The requester object that has to be notified
 *                      when something happens with the reservation.
 *                      Can be null.
 * @return              Reference to a new QoSSessionUnit object with
 *                      as many elements as requested. In the case of a
 *                      bi-directional session, there are two request
 *                      elements.
 */
QoSSessionUnit regularMultipleRequest (
    in RegularMultipleSpec requestSpec,
    in QoSRequester        requester)
    raises (APIException);


/* --- Regular Group Request --- */

typedef sequence<RegularMultipleSpec> RegularSpecSeq;

/**
 * Regular request for a group of QoS sessions,
 * e.g. for different service components per application.
 *
 * @param  groupName     The optional name of the new group (can be "").
 * @param  requestSpecs  An array of (multiple) request specifications.
 * @param  requester     The requester object that has to be notified
 *                       when something happens with the reservation.
 *                       Can be null.
 * @return               An new QoSSessionGroup object, containing
 *                       one QoSSessionUnit object for each (multiple)
 *                       request.
 */
QoSSessionGroup regularGroupRequest (
    in string         groupName,
    in RegularSpecSeq requestSpecs,
    in QoSRequester requester)
    raises (APIException);


/* --- Common --- */

/**
 * Creates an empty QoSSessionGroup.
 *
 * @param  groupName  The optional name of the new group (can be "").
 * @param  requester  The requester object that has to be notified
 *                    when something happens with the reservation.
 *                    Can be null.
 * @return            An new, empty QoSSessionGroup object.
 */
QoSSessionGroup createEmptyGroup (
    in string       groupName,
    in QoSRequester requester)
    raises (APIException);

/**
 * Returns all active reservations.
 *
 * @return  Sequence of QoSSession objects, both groups and units.
 */
QoSSessionSeq getActiveQoSSessions ();
```

```
        /**
         * Returns all active session units, also those which are in goups.
         *
         * @return  Sequence of QoSSessionUnit objects (no groups).
         */
        QoSSessionUnitSeq getQoSSessionUnits ();

        /**
         * Retrives the list of all former and actual reservations of this user.
         *
         * @return  Sequence of ReservationData incl. reservation parameters and
         *          accounting data.
         */
        eatPersistence::ReservationDataSeq retrieveReservationHistory ();

        /**
         * Closes explicitely a client session without logout;
         * removes the client session id from the list.
         *
         * @param   clientSessionId  Web servers' session id, included in the
         *                           URL.
         */
        void close (
            in string clientSessionId)
            raises (APIException);

        /**
         * Logs the end-user out, releases all reservations.
         */
        void logout ()
            raises (APIException);
    };


    /* --- QoS Session ------------------------------------------------------ */

    /**
     * SessionStatus indicates whether a requested (and by the EAT accepted)
     * reservation is still provisional (waiting for Proxy response) or already
     * accepted by the ACA and therefore active.
     * (Rejected reservations are immediately released, and the client is
     * informed.)
     */
    enum SessionStatus {
        Provisional,      // Waiting for the answer from the Proxy
        Active,           // Requested and admitted by the ACA
        Enabled           // Enabled at the edge router
    };

    /**
     * A QoSSession can be session group or unit.
     */
    interface QoSSession {

        /**
         * The unique session id of this session group or unit.
         */
        readonly attribute long sessionId;

        /**
         * Is this a session group or unit?
         */
        readonly attribute boolean isGroup;

        /**
         * The aggregating group. Can be null.
```

```
         */
        readonly attribute QoSSessionGroup group;

        /**
         * The requester of this session, can be null.
         */
        readonly attribute QoSRequester requester;

        /**
         * Releases the associated reservation unit (and all of its elements) or
         * the whole group (all aggregated units and groups are also released).
         * Automatically retrieves and stores the accounting data from the ACA
         * for each unit and all of its elements.
         */
        void release ()
            raises (APIException);

        /**
         * Suspends the associated reservation unit or the whole group,
         * i.e. all reservation units are released but without retrieving
         * any accounting information.
         */
        void suspend ()
            raises (APIException);
    };

    /**
     * The QoSSessionGroup interface belongs to a reservation group,
     * e.g. reservations (units) for several service components per application.
     */
    interface QoSSessionGroup : QoSSession {

        /**
         * The optional name of this group.
         */
        attribute string groupName;

        /**
         * The QoS sessions. Can be either other essions groups or units or
         * even both.
         */
        readonly attribute QoSSessionSeq sessions;

        /**
         * Returns one specific QoSSession object of this group.
         *
         * @param  sessionId  The session id of the reservation.
         * @return            The QoSSession object with the specified id.
         *                    Null if none.
         */
        QoSSession getSession (in long sessionId);

        /**
         * Adds/joins an already established QoSSession to this group.
         * Unlinks it from the former group/container.
         *
         * @param  sessionId  The session to be added to this group.
         */
        void join (in QoSSession session)
            raises (APIException);

        /**
         * Removes a QoSSession object from this group without releasing it.
         * Links it directly to the QoSSessionRequest.
         *
         * @param  sessionId  The session to be removed from this group.
         */
```

```
        void leave (in QoSSession session)
            raises (APIException);

        /**
         * Reamoves all QoSSession objects from this group without releasing
         * them.
         */
        void leaveAll ()
            raises (APIException);
    };

    /**
     * The QoSSessionUnit interface belongs to an actual reservation unit
     * (in terms of the ACA: a reservation bundle). While such a unit may
     * consist of more than one reservation element, it must be handled as only
     * one (multiple) reservation, for example a bi-directional one.
     */
    interface QoSSessionUnit : QoSSession {

        readonly attribute short numberOfElements; // e.g. 1 for an uni-
                                                   // directional reservation,
                                                   // 2 for a bidirectional one
        readonly attribute SessionStatus status;

        // Content of the advanced/regular request:
        readonly attribute appProfile::ProfileID applicationProfile;//"" if none
        readonly attribute string                applicationId;
        readonly attribute converter::QoSOption  selection;       //null if none
        readonly attribute ServiceIDSeq          networkServices;
        readonly attribute slsSeq                reqSLSs;
        readonly attribute long                  proxyId;

        /**
         * Gets the accounting data of this session.
         *
         * @return  Accounting information array of this session unit,
         *          containing one entry per requested reservation element.
         */
        aca::AccountingSeq getAccountingInformation ();

        /**
         * Enables an already active but not in the edge router installed
         * reservation.
         */
        void enable ()
            raises (APIException);
    };


    /* --- RequestEvent (for QoSRequester) ---------------------------------- */

    /**
     * RequestKind describes what with a reservation (request) ca be happen:
     * A provisional reservation can be accepted or rejected.
     * An already established reservation can be released, e.g. when it brokes
     * or when the schedule finishes.
     */
    enum RequestKind {
        Accepted,
        Rejected,
        Released
    };

    /**
     * RequestEvent contains the QoSSession id, the kind of the occuring event,
     * and a optional reason.
     */
```

```
valuetype RequestEvent : event::GeneralEvent {
    public long        sessionId;
    public RequestKind kind;
    public string      reason;
};


/* --- Application Manager ------------------------------------------- */

/**
 * ApplicationInformation contains the ID of the associated app profile,
 * the app's name, the version, and the build no.
 */
struct ApplicationInformation {
    appProfile::ProfileID applicationProfileID;
    string                applicationName;
    string                versionNo;
    string                buildNo;
    string                type;
    string                scope;
};
typedef sequence<ApplicationInformation> ApplicationInformationSeq;

/**
 * Semantical Group
 */
struct Semantical {
    string           description;
    string           type;
    string           language;
    sequence<string> qualifiers;
};
typedef sequence<Semantical> SemanticalSeq;

/**
 * Session Characteristics
 */
struct SessionCharacteristics {
    string        name;
    SemanticalSeq semantics;
};
typedef sequence<SessionCharacteristics> SessionCharacteristicsSeq;

/**
 * Service Component Option
 */
struct SCOption {
    string                    optionId;
    string                    description;
    string                    networkSpeed;
    string                    transportProtocol;
    SessionCharacteristicsSeq sessionCharacteristics;
};
typedef sequence<SCOption> SCOptionSeq;

/**
 * ServiceComponentInformation contains the ID of the associated sc profile,
 * the type and the options incl. the session characteristics.
 */
struct ServiceComponentInformation {
    appProfile::ProfileID serviceComponentProfileID;
    string                componentName;
    string                type;
    SCOptionSeq           options;
};
typedef sequence<ServiceComponentInformation> ServiceComponentInformationSeq;
```

```
/**
 * The ApplicationManager interface provides information about installed
 * available application (profiles), application proxies, etc.
 */
interface ApplicationManager {

    const string SERVER_PREFIX = "rcl.directory/eat.directory/api.directory/";
    const string SERVER_SUFFIX = ".ApplicationManager";

    /**
     * Gets all available apps for the Legacy App GUI.
     *
     * @return  Sequence of ApplicationInformation objects.
     */
    ApplicationInformationSeq getAvailableApps ();

    /**
     * Gets all available service components of one app profile.
     *
     * @return  Sequence of ServiceComponentInformation objects.
     */
    ServiceComponentInformationSeq getAvailableSCs (
        in appProfile::ProfileID applicationProfileID);

    /**
     * Any client, e.g. the servlets that construct the Regular Reservation
     * GUI may call this function to retrieve the OBJECTS representing the
     * xml profile. (See also appProfile::ProfileManager.)
     *
     * @param   applicationProfileID  The id of the Application Profile
     *
     * @return                        The reference to the profile object
     */
    appProfile::ExtApplicationProfile getApplicationProfile (
        in appProfile::ProfileID applicationProfileID)
        raises (APIException);

    /**
     * Any client, e.g. the servlets that construct the Regular Reservation
     * GUI may call this function to retrieve the STREAM representing the
     * xml profile. The servlets are responsible to internally parse the
     * profiles in order to retrieve the required information.
     * Due to the fact that the Application Profiles only refers to separate
     * Service Component Profiles, this operation has to be called several
     * times in order to get all data.
     *
     * @param   profile  The id of the Application Profile
     *
     * @return           The (byte) stream of the xml profile
     */
    XMLStream getProfileStream (
        in appProfile::ProfileID profile)
        raises (APIException);

    /**
     * Gets all installed proxies for the Reservation GUI.
     *
     * @return  Sequence of ApplicationProxy objects.
     */
    proxy::ProxyDescriptionSeq getApplicationProxies ();
};


/* --- Network Service Distributor ------------------------------------ */

/**
 * NetworkService is an "mirror" of the Network Service Profile, accessible
```

```
     * via CORBA. (An Application does not have access to the LDAP DB.)
     */
    struct NetworkServiceInformation {
        service::ServiceID          servId;
        string                      fullName;
    };
    typedef sequence<NetworkServiceInformation> NetworkServiceInformationSeq;

    /**
     * The ServiceDistributor interface provides information about available
     * network services, and which ones are included in the current SLA.
     */
    interface ServiceDistributor {

        const string SERVER_PREFIX = "rcl.directory/eat.directory/api.directory/";
        const string SERVER_SUFFIX = ".ServiceDistributor";

        /**
         * Gets all available network services.
         *
         * @return  Sequence of NetworkService objects.
         */
        NetworkServiceInformationSeq getNetworkServices ();

        /**
         * Any client, e.g. the servlets that construct the Regular Reservation
         * GUI may call this function to retrieve the OBJECTS representing the
         * xml data. (See also service::ServiceManager, and service.dtd.)
         *
         * @param   netService  The id of the network service
         *
         * @return              The reference to the network service object
         */
        service::NetworkService getNetworkService (
            in service::ServiceID netService)
            raises (APIException);
    };
};

#endif // api_idl
```

Application Profile Specification

The Application Profile description syntax has a new structure compared to the version proposed in [D2203]. For better readability and maintenance the app profile has been split in two "sub-profiles": ApplicationProfile.dtd, which refers to ServiceComponentProfile.dtd.

Firstly, the **ApplicationProfile.dtd** is shown:

```
<!--
     * Title:       ApplicationProfile
     * Description: The ApplicationProfile gives the possibility to
                    describe an application in detail in the scope of
                    QoS mechanisms (QoS offer and request) towards a
                    QoS enabled network. An ApplicationProfile
                    collects protocol information (to know which
                    ports are used, etc.) and implementation
                    information especially codec, and service
                    component information.
                    The implementation information describes the
                    QoSRequirements of the codec/service components,
                    the traffic produced by the codec, and gives
```

```
                    user-friendly descriptions of the different
                    possible quality levels.
                    This information enables:
                         1. the presentation towards the end-user of
                         the application quality levels
                         2. the request for QoS (QoS requirements and
                         produced traffic behaviour)
                         3. the connection to the network layer of the
                         QoS enables network
     * Copyright:    Copyright(c) Anne Thomas
     * Company:      TU Dresden
     * @author:      Anne Thomas
     * @E-mail:      Anne.Thomas@inf.tu-dresden.de
     * @version:     V11 - 29/05/2002
     */
 -->
<!ELEMENT ApplicationProfile (Implementation+, protocol*)>
<!ATTLIST ApplicationProfile
 name CDATA #REQUIRED
 version CDATA #REQUIRED
 build CDATA #IMPLIED
 type (VoIP | MULTIMEDIA | STREAMINGVIDEO | STREAMINGAUDIO | STREAMING
 | GAME | OTHER) #REQUIRED
 scope (unidirectional | bidirectional | p2p |
 xdirectional) #REQUIRED
>

<!ELEMENT Implementation (ServiceComponent,TransportProtocol*)>
<!--
    Applications can implement for their service components standard
    codecs. Nevertheless they interpret the codecs and the produced
    traffic for example depends directly of the concrete
    implementation. Here it is possible to reference application
    specific ServiceComponentProfiles mostly based on codecs and
    defining quality levels.
    As applications do not allways support all the quality levels of
    the ServiceComponent, a reference to the optionID of the
    ServiceComponent is necessary.
    !Note that if all optionIDs are implemented, no optionID
    references are necessary.
    A transport protocol is associated to each implemented
    ServiceComponent.
 -->
    <!ELEMENT ServiceComponent (name, optionID*)>
    <!ATTLIST ServiceComponent
        file CDATA #REQUIRED
    >
        <!ELEMENT optionID (#PCDATA)>
        <!ELEMENT name (#PCDATA)>
    <!ELEMENT TransportProtocol (lowerPortNo?,upperPortNo?)>
    <!ATTLIST TransportProtocol
         name (UNSPECIFIED | TCP | UDP) "TCP"
    >
<!ELEMENT protocol (lowerPortNo?,upperPortNo?,isControlPort?)>
<!ATTLIST protocol
 name (UNSPECIFIED | RTP | RTSP | RSVP | SIP | SDP | H320 | H321 |
 H322 | H323 | H324) "H323"
>
    <!ELEMENT isControlPort (#PCDATA)>
    <!ATTLIST isControlPort
     value (true | false) "false"
    >

<!ELEMENT lowerPortNo (#PCDATA)>
<!ATTLIST lowerPortNo
 value (fixed | configurable) "fixed"
>
```

```
<!ELEMENT upperPortNo (#PCDATA)>
<!ATTLIST upperPortNo
 value (fixed | configurable) "fixed"
>
```

With this `ApplicationProfile` syntax it is possible to:

- Describe the protocol used by an application by applying the `protocol` syntax.
- Describe the different service components composing an application by applying the `ServiceComponent` syntax.
- Describe the transport protocol of each service component by applying the `TransportProtocol` syntax.
- Describe each service component by applying the `ServiceComponentProfile` syntax.

Secondly, the **ServiceComponentProfile.dtd** is shown:

```
<!--
     * Title:       ServiceComponentProfile
     * Description: The ServiceComponentProfile gives the possibility
                    to specify service components in the scope of a
                    QoS request towards a QoS enabled transport
                    network. The ServiceComponentProfile offers
                    different quality levels for single service
                    components like voice, video etc. Therefor beside
                    the specification the QoS requirement attributes
                    in a generic way, it is possible to describe the
                    offered quality options in a first step at
                    end-user level, in a second step at network level.
                    The QoS attributes are extended with weights in
                    order to precise the requirements.
                    Except the AQUILASpecification needed for the
                    project, every other component of the
                    specification is generic.
     * Copyright:   Copyright(c) Anne Thomas
     * Company:     TU Dresden
     * @author:     Anne Thomas
     * @E-mail:     Anne.Thomas@inf.tu-dresden.de
     * @version:    29/05/2002 V1 - with weights
     */
 -->
<!ELEMENT ServiceComponentProfile  (QoSRequirement, Option+)>

<!ATTLIST ServiceComponentProfile
    name CDATA #REQUIRED
    serviceComponent (AUDIO | SPEECH | VIDEO | DATA | OTHER) #REQUIRED
>
<!--
    The QoSRequirement part of the ServiceComponentProfile corresponds
    to the general QoS requirements of the service component under
    whose the service component can work properly.
 -->
<!ELEMENT QoSRequirement (maxDelay, maxJitter, maxLoss, bwGuarantee,
ordering)>
<!--
    maxDelay    : "one way latency as unit milliseconds"
    maxJitter   : "delay variation as unit milliseconds"
```

```
    maxLoss     : "packet loss probability as unit percent"
    bwGuarantee : "percentage of bandwidth that is guaranteed"
    ordering    : "Must the packets be ordered?"
-->
    <!ELEMENT maxDelay (#PCDATA)>
    <!ATTLIST maxDelay
        unit CDATA #FIXED "ms"
        requirement (veryLow | low | medium | high | veryHigh
        | notRelevant) "medium"
        weight (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) "5"
    >
    <!ELEMENT maxJitter (#PCDATA)>
    <!ATTLIST maxJitter
        unit CDATA #FIXED "ms"
        requirement (veryLow | low | medium | high | veryHigh
        | notRelevant) "medium"
        weight (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) "5"
    >
    <!ELEMENT maxLoss (#PCDATA)>
    <!ATTLIST maxLoss
        unit CDATA #FIXED "percent"
        requirement (veryLow | low | medium | high | veryHigh
        | notRelevant) "medium"
        weight (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) "5"
    >
    <!ELEMENT bwGuarantee (#PCDATA)>
    <!ATTLIST bwGuarantee
        unit CDATA #FIXED "percent"
        requirement (veryLow | low | medium | high |
        veryHigh | notRelevant) "medium"
        weight (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) "5"
    >
    <!ELEMENT ordering EMPTY>
    <!ATTLIST ordering
        requirement (true | false) "true"
        weight (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) "5"
    >
<!--
    An Option corresponds to a "quality" level that is proposed by
    the implementation of the service compoenent, e.g. picture  size,
    sound quality. An Option consists of the generic so called
    SessionCharacteristics (the characteristics in the end-user
    language) and TrafficSpecification (describing the traffic
    produced by the application in qualitative and  quantitative
    terms, and the AQUILA dependent AQUILASpecification (technical
    information related to the NSs)
 -->
<!ELEMENT Option (SessionCharacteristic+, TrafficSpecification?,
AQUILASpecification?)>
<!--
    optionID    : explicit identifier of an option. In a service
                  component profile optionIDs should be unique.
    description : adjective, terms describing the service
                  component (e.g. medium, 4CIF ...)
    NetworkSpeed: explicit identifier of a network speed / end-user
                  equipment reference.  This NetworkSpeed attribut is
                  used as an ID that is aimed to be used in
                  correlation with the End-User Profile. Some
                  applications like NetMeeting adapt their traffic to
                  the end-user equipment. Therefore it is necessary in
                  the ServiceComponentProfile to refer to this
                  characteristic of the end-user equipment.
                  The description of the equipment of the end-user
                  profile should be the same as this NetworkSpeed
                  identifier.
-->
<!ATTLIST Option
```

```
    optionID CDATA #REQUIRED
    description CDATA #IMPLIED
    NetworkSpeed CDATA #IMPLIED
    TransportProtocol (UNSPECIFIED | TCP | UDP) "TCP"
>
    <!ELEMENT SessionCharacteristic (name, semanticalGroup*)>
    <!--
        name            : of the service component (e.g. picture size)
        semanticalGroup : semantical group for the service component
                          (e.g. user friendly description, in english)
        qualifier       : adjective, terms describing the service
                          component (e.g. medium, 4CIF ...)
     -->
        <!ELEMENT name (#PCDATA)>
        <!ELEMENT semanticalGroup (description, qualifier*)>
        <!ATTLIST semanticalGroup
            type (Technical | UserFriendly) "UserFriendly"
            language (en | de | po | fr | it | es | fi | gr) "en"
        >
            <!ELEMENT qualifier (#PCDATA)>
    <!ELEMENT TrafficSpecification  (type+, duration, adaptivity,
    burstiness, packetSize, bitRate, flow)>
    <!--
        type       : "type of the traffic"
        duration   : "living time of the traffic"
        adaptivity : "adaptivity of the traffic to the capacity of the
                      connection (application level QoS adaptation)"
        burstiness : "burstiness of the traffic"
        packetSize : "size ofthe packet"
        bitRate    : "bit rate"
        flow       : "greediness of the flow"
    -->
        <!ELEMENT type EMPTY>
        <!ATTLIST type
            type (realTime | nonRealTime | stream |
            elastic) "nonRealTime"
        >
        <!ELEMENT duration EMPTY>
        <!ATTLIST duration
            value (shortLiving | longLiving )
            "shortLiving"
        >
        <!ELEMENT adaptivity EMPTY>
        <!ATTLIST adaptivity
            value (true | false) "false"
        >
        <!ELEMENT burstiness EMPTY>
        <!ATTLIST burstiness
            value (true | false) "false"
        >
        <!ELEMENT packetSize (averagePacketSize?, maximumPacketSize?,
        minimumPolicedUnit?)>
        <!ATTLIST packetSize
            variability (constant | variable) "constant"
            qualitatively (verySmall | small | medium | big
            | veryBig) "medium"
        >
            <!ELEMENT averagePacketSize (#PCDATA)>
            <!ATTLIST averagePacketSize
                unit CDATA #FIXED "bytes"
                qualitatively (verySmall | small | medium | big
                | veryBig) "medium"
            >
            <!ELEMENT maximumPacketSize (#PCDATA)>
            <!ATTLIST maximumPacketSize
                unit CDATA #FIXED "bytes"
                qualitatively (verySmall | small | medium | big
```

```
                | veryBig) "medium"
            >
            <!ELEMENT minimumPolicedUnit (#PCDATA)>
            <!ATTLIST minimumPolicedUnit
                unit CDATA #FIXED "bytes"
                qualitatively (verySmall | small | medium | big
                | veryBig) "medium"
            >
        <!ELEMENT bitRate (peakRate?, averageRate)>
        <!ATTLIST bitRate
            variability (constant | variable) "constant"
            qualitatively (veryLow | low | medium | high |
            veryHigh) "medium"
        >
            <!ELEMENT peakRate (#PCDATA)>
            <!ATTLIST peakRate
                unit CDATA #FIXED "bit/s"
                qualitatively (veryLow | low | medium | high |
                veryHigh) "medium"
            >
            <!ELEMENT averageRate (#PCDATA)>
            <!ATTLIST averageRate
                unit CDATA #FIXED "bit/s"
                qualitatively (veryLow | low | medium | high |
                veryHigh) "medium"
            >
        <!ELEMENT flow EMPTY>
        <!ATTLIST flow
            value (greedy | non-greedy) "non-greedy"
        >

    <!ELEMENT AQUILASpecification (serviceID, BSP, BSS, minPU, maxPS,
    PR, SR)>
    <!--
        serviceID : Name of the AQUILA NS
        BSP       : bucket size for PR (bytes)
        BSS       : bucket size for SR (bytes)
        minPU     : minimum policed unit (bytes)
        maxPS     : maximum (allowed) packet size (bytes)
        PR        : peak rate (bit/s)
        SR        : sustainable rate (bit/s)
        EAR       : Expected Average Rate (bit/s) - Not used
        PR1       : first threshold for bilevel (bit/s) - Not used
        PR2       : second threshold for bilevel (bit/s) - Not used
    -->
        <!ELEMENT serviceID EMPTY>
        <!ATTLIST serviceID
            value (PCBR | PVBR | PMM | PMC | STD | CUSTOM) "STD"
        >
        <!ELEMENT BSP (#PCDATA)>
        <!ATTLIST BSP
            unit CDATA #FIXED "bytes"
        >
        <!ELEMENT BSS (#PCDATA)>
        <!ATTLIST BSS
            unit CDATA #FIXED "bytes"
        >
        <!ELEMENT minPU (#PCDATA)>
        <!ATTLIST minPU
            unit CDATA #FIXED "bytes"
        >
        <!ELEMENT maxPS (#PCDATA)>
        <!ATTLIST maxPS
            unit CDATA #FIXED "bytes"
        >
        <!ELEMENT PR (#PCDATA)>
        <!ATTLIST PR
```

```
            unit CDATA #FIXED "bit/s"
        >
        <!ELEMENT SR (#PCDATA)>
        <!ATTLIST SR
            unit CDATA #FIXED "bit/s"
        >
<!ELEMENT description (#PCDATA)>
```

With the `ServiceComponentProfile` syntax it is possible to:

- Describe the different possible QoS options of a service component by applying the `Option` syntax.
- Describe the QoS expectations of the service components by applying the `QoSRequirement` syntax.
- Describe the traffic produced by a service component by applying the `TrafficSpecification` syntax.
- Describe application characteristics at end-user level in a user-friendly manner by applying the `SessionCharacteristic` syntax.

Application Profile Library

In the following the three Application Profiles as well as the four Service Component Profiles for the CIS Mediazine are shown. They build the basis for the profile library to be extended later on:

Application Profile for RealSystem

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ApplicationProfile PUBLIC "RealSystem 8.5"
"../dtd/ApplicationProfileV11.dtd">
<ApplicationProfile type="STREAMINGVIDEO" version="8.5" name="RealSystem"
scope="unidirectional">
    <Implementation>
        <ServiceComponent file="RealSystem_8.5_Video_v01.xml">
            <name>RealSystem_8.5_Video_v01</name>
        </ServiceComponent>
        <TransportProtocol name="TCP">
            <lowerPortNo>554</lowerPortNo>
            <upperPortNo>554</upperPortNo>
        </TransportProtocol>
        <TransportProtocol name="UDP">
            <lowerPortNo>6970</lowerPortNo>
            <upperPortNo>6970</upperPortNo>
        </TransportProtocol>
    </Implementation>
    <Implementation>
        <ServiceComponent file="RealSystem_8.5_Audio_v01.xml">
            <name>RealSystem_8.5_Audio_v01</name>
        </ServiceComponent>
        <TransportProtocol name="TCP">
            <lowerPortNo>554</lowerPortNo>
            <upperPortNo>554</upperPortNo>
```

```
        </TransportProtocol>
        <TransportProtocol name="UDP">
            <lowerPortNo>6970</lowerPortNo>
            <upperPortNo>6970</upperPortNo>
        </TransportProtocol>
    </Implementation>
    <protocol name="RTSP">
        <isControlPort value="false"/>
    </protocol>
</ApplicationProfile>
```

Service Component Profile VIDEO for RealSystem

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ServiceComponentProfile PUBLIC "RealSystem_8.5_Video_v01"
"../dtd/ServiceComponentProfileV1.dtd">
<ServiceComponentProfile name="RealSystem_8.5_Video_v01" serviceComponent="VIDEO">
    <QoSRequirement>
        <maxDelay unit="ms" requirement="veryLow" weight="2">1000</maxDelay>
        <maxJitter unit="ms" requirement="notRelevant" weight="0">500</maxJitter>
        <maxLoss unit="percent" requirement="medium" weight="5">5</maxLoss>
        <bwGuarantee unit="percent" requirement="high" weight="5">90</bwGuarantee>
        <ordering weight="5" requirement="false"/>
    </QoSRequirement>
    <Option optionID="1" description="video low quality scenario"
NetworkSpeed="56kBit/s Modem / ISDN / Dual ISDN" TransportProtocol="UDP">
        <SessionCharacteristic>
            <name>video quality</name>
            <semanticalGroup type="UserFriendly" language="en">
                <description>Video quality</description>
                <qualifier>very low quality (45kBit/s)</qualifier>
            </semanticalGroup>
            <semanticalGroup type="UserFriendly" language="de">
                <description>Video-Qualität</description>
                <qualifier>sehr niedrige Qualität (45kBit/s)</qualifier>
            </semanticalGroup>
        </SessionCharacteristic>
        <TrafficSpecification>
            <type type="stream"/>
            <duration value="longLiving"/>
            <adaptivity value="true"/>
            <burstiness value="false"/>
            <packetSize qualitatively="small" variability="variable">
                <averagePacketSize qualitatively="medium" unit="bytes"/>
                <maximumPacketSize qualitatively="veryBig" unit="bytes"/>
                <minimumPolicedUnit qualitatively="medium" unit="bytes"/>
            </packetSize>
            <bitRate qualitatively="low" variability="variable">
                <peakRate qualitatively="low" unit="bit/s"/>
                <averageRate qualitatively="low" unit="bit/s"/>
            </bitRate>
            <flow value="non-greedy"/>
        </TrafficSpecification>
        <AQUILASpecification>
            <serviceID value="PMM"/>
            <BSP unit="bytes">56000</BSP>
            <BSS unit="bytes">5625</BSS>
            <minPU unit="bytes">1</minPU>
            <maxPS unit="bytes">1500</maxPS>
            <PR unit="bit/s">56000</PR>
            <SR unit="bit/s">46080</SR>
        </AQUILASpecification>
    </Option>
    <Option optionID="2" description="video medium quality scenario"
NetworkSpeed="Cable Modem / DSL" TransportProtocol="UDP">
```

```
        <SessionCharacteristic>
            <name>video quality</name>
            <semanticalGroup type="UserFriendly" language="en">
                <description>Video quality</description>
                <qualifier>medium quality (384kBits/s)</qualifier>
            </semanticalGroup>
            <semanticalGroup type="UserFriendly" language="de">
                <description>Video-Qualität</description>
                <qualifier>mittlere Qualität (384kBit/s)</qualifier>
            </semanticalGroup>
        </SessionCharacteristic>
        <TrafficSpecification>
            <type type="stream"/>
            <duration value="longLiving"/>
            <adaptivity value="true"/>
            <burstiness value="false"/>
            <packetSize qualitatively="medium" variability="variable">
                <averagePacketSize qualitatively="medium" unit="bytes"/>
                <maximumPacketSize qualitatively="veryBig" unit="bytes"/>
                <minimumPolicedUnit qualitatively="medium" unit="bytes"/>
            </packetSize>
            <bitRate qualitatively="medium" variability="variable">
                <peakRate qualitatively="low" unit="bit/s"/>
                <averageRate qualitatively="low" unit="bit/s"/>
            </bitRate>
            <flow value="non-greedy"/>
        </TrafficSpecification>
        <AQUILASpecification>
            <serviceID value="PMM"/>
            <BSP unit="bytes">480000</BSP>
            <BSS unit="bytes">49152</BSS>
            <minPU unit="bytes">7</minPU>
            <maxPS unit="bytes">1500</maxPS>
            <PR unit="bit/s">480000</PR>
            <SR unit="bit/s">393216</SR>
        </AQUILASpecification>
    </Option>
    <Option optionID="3" description="video high quality scenario"
NetworkSpeed="LAN" TransportProtocol="UDP">
        <SessionCharacteristic>
            <name>video quality</name>
            <semanticalGroup type="UserFriendly" language="en">
                <description>Video quality</description>
                <qualifier>high quality (1024kBits/s)</qualifier>
            </semanticalGroup>
            <semanticalGroup type="UserFriendly" language="de">
                <description>Video-Qualität</description>
                <qualifier>hohe Qualität (1024kBit/s)</qualifier>
            </semanticalGroup>
        </SessionCharacteristic>
        <TrafficSpecification>
            <type type="stream"/>
            <duration value="longLiving"/>
            <adaptivity value="true"/>
            <burstiness value="false"/>
            <packetSize qualitatively="big" variability="variable">
                <averagePacketSize qualitatively="big" unit="bytes"/>
                <maximumPacketSize qualitatively="veryBig" unit="bytes"/>
                <minimumPolicedUnit qualitatively="medium" unit="bytes"/>
            </packetSize>
            <bitRate qualitatively="high" variability="variable">
                <peakRate qualitatively="low" unit="bit/s"/>
                <averageRate qualitatively="low" unit="bit/s"/>
            </bitRate>
            <flow value="non-greedy"/>
        </TrafficSpecification>
        <AQUILASpecification>
```

```
            <serviceID value="PMM"/>
            <BSP unit="bytes">1280000</BSP>
            <BSS unit="bytes">131072</BSS>
            <minPU unit="bytes">19</minPU>
            <maxPS unit="bytes">1500</maxPS>
            <PR unit="bit/s">1280000</PR>
            <SR unit="bit/s">1048576</SR>
        </AQUILASpecification>
    </Option>
</ServiceComponentProfile>
```

Service Component Profile AUDIO for RealSystem

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ServiceComponentProfile PUBLIC "RealSystem_8.5_Audio_v01"
"../dtd/ServiceComponentProfileV1.dtd">
<ServiceComponentProfile name="RealSystem_8.5_Audio_v01" serviceComponent="AUDIO">
    <QoSRequirement>
        <maxDelay unit="ms" requirement="veryLow" weight="2">1000</maxDelay>
        <maxJitter unit="ms" requirement="notRelevant" weight="0">500</maxJitter>
        <maxLoss unit="percent" requirement="medium" weight="5">5</maxLoss>
        <bwGuarantee unit="percent" requirement="high" weight="5">90</bwGuarantee>
        <ordering weight="5" requirement="false"/>
    </QoSRequirement>
    <Option optionID="1" description="audio low quality scenario"
NetworkSpeed="56kBit/s Modem / ISDN" TransportProtocol="UDP">
        <SessionCharacteristic>
            <name>audio quality</name>
            <semanticalGroup type="UserFriendly" language="en">
                <description>Audio quality</description>
                <qualifier>very low quality (48kBit/s)</qualifier>
            </semanticalGroup>
            <semanticalGroup type="UserFriendly" language="de">
                <description>Audio-Qualität</description>
                <qualifier>sehr niedrige Qualität (48kBit/s)</qualifier>
            </semanticalGroup>
        </SessionCharacteristic>
        <TrafficSpecification>
            <type type="stream"/>
            <duration value="longLiving"/>
            <adaptivity value="true"/>
            <burstiness value="false"/>
            <packetSize qualitatively="small" variability="constant">
                <averagePacketSize qualitatively="medium" unit="bytes"/>
                <maximumPacketSize qualitatively="veryBig" unit="bytes"/>
                <minimumPolicedUnit qualitatively="medium" unit="bytes"/>
            </packetSize>
            <bitRate qualitatively="low" variability="constant">
                <peakRate qualitatively="low" unit="bit/s"/>
                <averageRate qualitatively="low" unit="bit/s"/>
            </bitRate>
            <flow value="non-greedy"/>
        </TrafficSpecification>
        <AQUILASpecification>
            <serviceID value="PMM"/>
            <BSP unit="bytes">60000</BSP>
            <BSS unit="bytes">6144</BSS>
            <minPU unit="bytes">1</minPU>
            <maxPS unit="bytes">1500</maxPS>
            <PR unit="bit/s">60000</PR>
            <SR unit="bit/s">49152</SR>
        </AQUILASpecification>
    </Option>
    <Option optionID="2" description="audio medium quality scenario"
NetworkSpeed="Dual ISDN" TransportProtocol="UDP">
```

```
        <SessionCharacteristic>
            <name>audio quality</name>
            <semanticalGroup type="UserFriendly" language="en">
                <description>Audio quality</description>
                <qualifier>medium quality (112kBits/s)</qualifier>
            </semanticalGroup>
            <semanticalGroup type="UserFriendly" language="de">
                <description>Audio-Qualität</description>
                <qualifier>mittlere Qualität (112kBit/s)</qualifier>
            </semanticalGroup>
        </SessionCharacteristic>
        <TrafficSpecification>
            <type type="stream"/>
            <duration value="longLiving"/>
            <adaptivity value="true"/>
            <burstiness value="false"/>
            <packetSize qualitatively="small" variability="constant">
                <averagePacketSize qualitatively="medium" unit="bytes"/>
                <maximumPacketSize qualitatively="veryBig" unit="bytes"/>
                <minimumPolicedUnit qualitatively="medium" unit="bytes"/>
            </packetSize>
            <bitRate qualitatively="low" variability="constant">
                <peakRate qualitatively="low" unit="bit/s"/>
                <averageRate qualitatively="low" unit="bit/s"/>
            </bitRate>
            <flow value="non-greedy"/>
        </TrafficSpecification>
        <AQUILASpecification>
            <serviceID value="PMM"/>
            <BSP unit="bytes">140000</BSP>
            <BSS unit="bytes">14336</BSS>
            <minPU unit="bytes">2</minPU>
            <maxPS unit="bytes">1500</maxPS>
            <PR unit="bit/s">140000</PR>
            <SR unit="bit/s">114688</SR>
        </AQUILASpecification>
    </Option>
    <Option optionID="3" description="audio high quality scenario"
NetworkSpeed="Cable Modem / DSL / LAN" TransportProtocol="UDP">
        <SessionCharacteristic>
            <name>audio quality</name>
            <semanticalGroup type="UserFriendly" language="en">
                <description>Audio quality</description>
                <qualifier>high quality (256kBits/s)</qualifier>
            </semanticalGroup>
            <semanticalGroup type="UserFriendly" language="de">
                <description>Audio-Qualität</description>
                <qualifier>hohe Qualität (256kBit/s)</qualifier>
            </semanticalGroup>
        </SessionCharacteristic>
        <TrafficSpecification>
            <type type="stream"/>
            <duration value="longLiving"/>
            <adaptivity value="true"/>
            <burstiness value="false"/>
            <packetSize qualitatively="medium" variability="constant">
                <averagePacketSize qualitatively="big" unit="bytes"/>
                <maximumPacketSize qualitatively="veryBig" unit="bytes"/>
                <minimumPolicedUnit qualitatively="medium" unit="bytes"/>
            </packetSize>
            <bitRate qualitatively="medium" variability="constant">
                <peakRate qualitatively="low" unit="bit/s"/>
                <averageRate qualitatively="low" unit="bit/s"/>
            </bitRate>
            <flow value="non-greedy"/>
        </TrafficSpecification>
        <AQUILASpecification>
```

```
            <serviceID value="PMM"/>
            <BSP unit="bytes">320000</BSP>
            <BSS unit="bytes">32768</BSS>
            <minPU unit="bytes">5</minPU>
            <maxPS unit="bytes">1500</maxPS>
            <PR unit="bit/s">320000</PR>
            <SR unit="bit/s">262144</SR>
        </AQUILASpecification>
    </Option>
</ServiceComponentProfile>
```

Application Profile for NetMeeting

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ApplicationProfile PUBLIC "NetMeeting 3.01"
"../dtd/ApplicationProfileV11.dtd">
<ApplicationProfile build="4.4.3388" type="MULTIMEDIA" version="3.01"
name="NetMeeting" scope="xdirectional">
    <Implementation>
        <ServiceComponent file="NetMeeting_3.01_Video_v01.xml">
            <name>NetMeeting_3.01_Video_v01</name>
        </ServiceComponent>
        <TransportProtocol name="TCP"/>
    </Implementation>
    <protocol name="H323">
        <isControlPort value="true">1720</isControlPort>
    </protocol>
    <protocol name="RTP">
        <isControlPort value="false"/>
    </protocol>
</ApplicationProfile>
```

Service Component Profile VIDEO for NetMeeting

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ServiceComponentProfile PUBLIC "NetMeeting_3.01_Video_v01"
"../dtd/ServiceComponentProfileV1.dtd">
<ServiceComponentProfile name="NetMeeting_3.01_Video_v01" serviceComponent="VIDEO">
    <QoSRequirement>
        <maxDelay unit="ms" requirement="high" weight="1">1200</maxDelay>
        <maxJitter unit="ms" requirement="low" weight="3">120</maxJitter>
        <maxLoss unit="percent" requirement="medium" weight="5">10</maxLoss>
        <bwGuarantee unit="percent" requirement="high" weight="8">0</bwGuarantee>
        <ordering weight="8" requirement="true"/>
    </QoSRequirement>
    <Option optionID="1" description="video low quality scenario"
NetworkSpeed="56kBit/s Modem" TransportProtocol="TCP">
        <SessionCharacteristic>
            <name>video quality</name>
            <semanticalGroup type="UserFriendly" language="en">
                <description>Video quality</description>
                <qualifier>very low quality (28.8kBit/s)</qualifier>
            </semanticalGroup>
            <semanticalGroup type="UserFriendly" language="de">
                <description>Video-Qualität</description>
                <qualifier>sehr niedrige Qualität (28.8kBit/s)</qualifier>
            </semanticalGroup>
        </SessionCharacteristic>
        <TrafficSpecification>
            <type type="elastic"/>
            <duration value="longLiving"/>
            <adaptivity value="false"/>
```

```
            <burstiness value="true"/>
            <packetSize qualitatively="medium" variability="variable">
                <averagePacketSize qualitatively="medium" unit="bytes"/>
                <maximumPacketSize qualitatively="medium" unit="bytes"/>
                <minimumPolicedUnit qualitatively="small" unit="bytes"/>
            </packetSize>
            <bitRate qualitatively="high" variability="variable">
                <peakRate qualitatively="medium" unit="bit/s"/>
                <averageRate qualitatively="medium" unit="bit/s"/>
            </bitRate>
            <flow value="greedy"/>
        </TrafficSpecification>
        <AQUILASpecification>
            <serviceID value="PVBR"/>
            <BSP unit="bytes">2000</BSP>
            <BSS unit="bytes">2048</BSS>
            <minPU unit="bytes">60</minPU>
            <maxPS unit="bytes">1500</maxPS>
            <PR unit="bit/s">28800</PR>
            <SR unit="bit/s">19200</SR>
        </AQUILASpecification>
    </Option>
    <Option optionID="2" description="video medium quality scenario"
NetworkSpeed="ISDN / Dual ISDN" TransportProtocol="TCP">
        <SessionCharacteristic>
            <name>video quality</name>
            <semanticalGroup type="UserFriendly" language="en">
                <description>Video quality</description>
                <qualifier>medium quality (64kBits/s)</qualifier>
            </semanticalGroup>
            <semanticalGroup type="UserFriendly" language="de">
                <description>Video-Qualität</description>
                <qualifier>mittlere Qualität (64kBit/s)</qualifier>
            </semanticalGroup>
        </SessionCharacteristic>
        <TrafficSpecification>
            <type type="elastic"/>
            <duration value="longLiving"/>
            <adaptivity value="false"/>
            <burstiness value="true"/>
            <packetSize qualitatively="medium" variability="variable">
                <averagePacketSize qualitatively="medium" unit="bytes"/>
                <maximumPacketSize qualitatively="big" unit="bytes"/>
                <minimumPolicedUnit qualitatively="medium" unit="bytes"/>
            </packetSize>
            <bitRate qualitatively="high" variability="variable">
                <peakRate qualitatively="high" unit="bit/s"/>
                <averageRate qualitatively="medium" unit="bit/s"/>
            </bitRate>
            <flow value="greedy"/>
        </TrafficSpecification>
        <AQUILASpecification>
            <serviceID value="PVBR"/>
            <BSP unit="bytes">2000</BSP>
            <BSS unit="bytes">4048</BSS>
            <minPU unit="bytes">60</minPU>
            <maxPS unit="bytes">1500</maxPS>
            <PR unit="bit/s">64000</PR>
            <SR unit="bit/s">48000</SR>
        </AQUILASpecification>
    </Option>
    <Option optionID="3" description="video high quality scenario"
NetworkSpeed="Cable Modem / DSL / LAN" TransportProtocol="TCP">
        <SessionCharacteristic>
            <name>video quality</name>
            <semanticalGroup type="UserFriendly" language="en">
                <description>Video quality</description>
```

```
                <qualifier>high quality (160kBits/s)</qualifier>
            </semanticalGroup>
            <semanticalGroup type="UserFriendly" language="de">
                <description>Video-Qualität</description>
                <qualifier>hohe Qualität (160kBit/s)</qualifier>
            </semanticalGroup>
        </SessionCharacteristic>
        <TrafficSpecification>
            <type type="elastic"/>
            <duration value="longLiving"/>
            <adaptivity value="false"/>
            <burstiness value="true"/>
            <packetSize qualitatively="big" variability="variable">
                <averagePacketSize qualitatively="big" unit="bytes"/>
                <maximumPacketSize qualitatively="veryBig" unit="bytes"/>
                <minimumPolicedUnit qualitatively="medium" unit="bytes"/>
            </packetSize>
            <bitRate qualitatively="high" variability="variable">
                <peakRate qualitatively="high" unit="bit/s"/>
                <averageRate qualitatively="high" unit="bit/s"/>
            </bitRate>
            <flow value="greedy"/>
        </TrafficSpecification>
        <AQUILASpecification>
            <serviceID value="PVBR"/>
            <BSP unit="bytes">2000</BSP>
            <BSS unit="bytes">5120</BSS>
            <minPU unit="bytes">60</minPU>
            <maxPS unit="bytes">1500</maxPS>
            <PR unit="bit/s">160000</PR>
            <SR unit="bit/s">75000</SR>
        </AQUILASpecification>
    </Option>
</ServiceComponentProfile>
```

Application Profile for OIDS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ApplicationProfile PUBLIC "OIDS-Game" "../dtd/ApplicationProfileV11.dtd">
<ApplicationProfile type="GAME" version="unknown" name="OIDS" scope="xdirectional">
    <Implementation>
        <ServiceComponent file="OIDS-Game_Data_v01.xml">
            <name>OIDS-Game_Data_v01</name>
        </ServiceComponent>
        <TransportProtocol name="TCP">
            <lowerPortNo>-1</lowerPortNo>
            <upperPortNo>-1</upperPortNo>
        </TransportProtocol>
    </Implementation>
</ApplicationProfile>
```

Service Component Profile DATA for OIDS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ServiceComponentProfile PUBLIC "OIDS-Game_v01"
"../dtd/ServiceComponentProfileV1.dtd">
<ServiceComponentProfile name="OIDS-Game_Data_v01" serviceComponent="DATA">
    <QoSRequirement>
        <maxDelay unit="ms" requirement="low" weight="8">100</maxDelay>
        <maxJitter unit="ms" requirement="low" weight="5">20</maxJitter>
        <maxLoss unit="percent" requirement="low" weight="8">2</maxLoss>
        <bwGuarantee unit="percent" requirement="high" weight="5">90</bwGuarantee>
```

```
        <ordering weight="7" requirement="true"/>
    </QoSRequirement>
    <Option optionID="1" description="QoS enabled" NetworkSpeed="56kBit/s Modem /
ISDN / Dual ISDN / Cable Modem / DSL /LAN" TransportProtocol="TCP">
        <SessionCharacteristic>
            <name>QoS</name>
            <semanticalGroup type="UserFriendly" language="en">
                <description>QoS</description>
                <qualifier>enabled</qualifier>
            </semanticalGroup>
            <semanticalGroup type="UserFriendly" language="de">
                <description>QoS</description>
                <qualifier>eingeschaltet</qualifier>
            </semanticalGroup>
        </SessionCharacteristic>
        <TrafficSpecification>
            <type type="realTime"/>
            <duration value="shortLiving"/>
            <adaptivity value="false"/>
            <burstiness value="false"/>
            <packetSize qualitatively="small" variability="constant">
                <averagePacketSize qualitatively="small" unit="bytes"/>
                <maximumPacketSize qualitatively="medium" unit="bytes"/>
                <minimumPolicedUnit qualitatively="medium" unit="bytes"/>
            </packetSize>
            <bitRate qualitatively="low" variability="constant">
                <peakRate qualitatively="medium" unit="bit/s"/>
                <averageRate qualitatively="low" unit="bit/s"/>
            </bitRate>
            <flow value="non-greedy"/>
        </TrafficSpecification>
        <AQUILASpecification>
            <serviceID value="PMC"/>
            <BSP unit="bytes">40000</BSP>
            <BSS unit="bytes">2048</BSS>
            <minPU unit="bytes">1</minPU>
            <maxPS unit="bytes">1500</maxPS>
            <PR unit="bit/s">40000</PR>
            <SR unit="bit/s">2048</SR>
        </AQUILASpecification>
    </Option>
</ServiceComponentProfile>
```