CHALMERS

Shared Resource for Collaborative Editing over a Wireless Network

Master of Science Thesis

JONAS ÅDAHL

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, December 2010

Shared Resource for Collaborative Editing over a Wireless Network

Jonas Ådahl

Examiner: Marina Papatriantafilou

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden December 2010

Shared Resource for Collaborative Editing over a Wireless Network
Jonas Ådahl
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

Today hand-held consumer devices such as Smartphones and Mobile Internet Devices are getting more and more powerful and feature rich. They are no longer only built with network capabilities through a cellular network such as GSM[1] or 3G[2], but also high speed Wireless Local Area Networks which opens up more possibilities.

While Internet based applications, such as Web browsers, E-mail clients, and Multimedia based applications have been the major part of the market, programs using properties of a distributed system are sparse.

This Thesis report describes the background, design choices and development of an application using distributed group communication techniques to create a collaborative work space on a typical Smartphone. It also shows how to use this to implementation a Whiteboard application. It explores the possibilites of using well known protocols, standards as well as Free Software for solving the various problems associated with these functions, such as presence discovery, group communication and whiteboarding.

Sammanfattning

Dagens bärbara handdatorer såsom Smartphones och mobila internetenheter blir hela tiden kraftigare och mer funktionsrika. De är inte längre endast byggda med nätverksmöjligheter via ett mobiltelefonnätet som till exempel GSM eller 3G, utan även också med lokala trådlösa höghastighetsnätverk vilket öppnar upp nya möjligheter.

Då internetbaserade applikationer, såsom webläsare och e-mailklienter, samt multimediabaserade applikationer har varit en stor del av marknaden, program som använder egenskaper hos ett distribuerat nätverk är mer ovanligt.

Det här examensarbetet beskriver bakgrunden, designval samt utveckling av en applikation som använder distribuerad gruppkommunikationstekniker för att skapa en kollaborativ arbetsyta på en typisk Smartphone. Det visas också hur detta går att använda för att implementera en Whiteboard-applikation. Möjligheterna att använda välkända

[1] Global System for Mobile communications (GSM) is the most used standard for mobile phone communication in the world. GSM is considered the second generation (2G)

[2] 3G is a family of standards for wireless communication designed to succeed 2G by delivering higher transfer rates.

protokoll, standarder samt fri mjukvara för att lösa de problem anknytna till funktionerna, såsom upptäcka närliggande enheter, gruppkommunikation och whiteboarding, utforskas i den här rapporten.

Preface

This thesis work was done as a part time research engineer at the Industrial Technology Research Institute (ITRI) in JhuDong Township, Taiwan. Special thanks to the people at my division (Advanced Technology Division) in the Information and Communication Laboratories and my supervisor at ITRI: Tan, Koan-Sin.

I would also like to thank my supervisor at Chalmers University of Technology, Marina Papatriantafilou.

Other people I would like to show my appreciation are Professor Paul Lin and Professor Wu, I-Chen for helping me finding my thesis position at ITRI; and Mr An, Hua-Cheng for helping me with administrative things regarding my foreign residence in Taiwan.

Contents

1 Introduction

The goal of this thesis project was to create a collaborative work space implemented on a modern Smartphone like device using a local area wireless network as communication medium. This means that there are a number of problems that needs to be considered. When developing on a Smartphone device battery power have to be considered, as well the limitations of the development environment; it is common that a Third-party developer is limited to only a subset of a popular programming language. Also, since the collaborative work space is supposed to rely only on a wireless network, not a central server, a group communication method has to be utilized. To implement and a kind of synchronization protocol has to be implemented to keep the state of the shared document equal between the devices. These issues are discussed separately in this report and several alternatives are given, as well as the design choice and implementation of an application.

There is a lot of previous work done concerning the given problems and this report gives a summary of various research projects and toolkits dealing these issues. It digs into the different protocols handling auto discovery and distribution of presences, the complications that comes with Group Communication such as membership and atomic reliable broadcast, and the difficulties involving collaborative editing, such as changing number of participants, net-splits, joining and leaving and keeping a synchronized state. Also previous projects and different possibilities to implement a shared Whiteboard are discussed.

2 Goals

While coming up with ahe thesis subject some certain goals were set up about what kind of project this would be. The most early idea was to implement a program that can collaboratively edit a whiteboard over a ad-hoc network using nothing more than today's smartphone devices. The procedure should be completely distributed in the way that no central point of synchronization was to be used and users can come and go as they like while the whiteboard still stays intact and synchronized between all active participants.

During the process of designing, the goals were altered in a way making them more reasonable. For example ad-hoc networking using smartphone devices would for the most part be in the scope the operating system of the smartphone rather than the application, thus that goal was simply changed to a possible future addition when such features are more common.

A set of goals are introduced in the following sections. The goals of the application in general, what is expected of the presence and one-on-one communication methods and last collaboration goals.

2.1 Application

An important goal of the application was, where possible, make the application as **portable** as possible; in other words, the implementation should not be limited to be compatible only with one platform, except for the user interface, which in many cases needs to be specific to a platform.

When designing the protocols and technologies used within the application, a goal was to make them **extendible** in a way that adding more functionality should be possible without breaking compatibility. It can be said that a problem should be solved in the most general way possible, so that other problems may fall under the same solution with no or few additions.

Lastly, a goal was to use and contribute to **open source/free software** and protocols whenever possible. See Section 3.5 for more discussion about this subject.

2.2 Presence and One-on-One Communication

Presence in a network can be designed to be completely service oriented, such as presences of printers in a network or similar. However, this application should be used with

a presence that is more **social** where social means it should be able to personalize a presence by changing mode, adding personal messages or similar.

Another goal of the presence problem was to make it **easy to use**. There should be no configuration required to be able to discover other users presences or advertise one's own presence.

A goal of the communication protocol was to make it **extensible**. A common problem with communication protocols is adding functionality to an existing one may break compatibility. However, protocols may be designed in ways making it extensible. It must not be limited to only be used to solve the problems of the application written about in this report.

2.3 Collaboration

The primary goal of the collaboration was that it should use a completely **decentralized** method for synchronization the state of the whiteboard between any number of users. Users should be able to leave and join, while the state of the whiteboard stays intact.

Another aim was to make the protocol as **general** as possible, not limiting the collaboration to be used only with whiteboards. It should be considered to be synchronization of a document, where the document may be of other types than a whiteboard. The primary document type however is to be some kind of whiteboard style document.

A goal, but not a requirement, was for the protocol used for synchronization should be usable over the Internet by using a multi user chat room style component.

3 Background

This chapter provides background information and related research about the various problems concerning the design and implementation of this application as well as hardware choice. The discussed problems are joining a network, finding remote presences and distributing own presences, group communication, collaboratively editing of the shared state, and lastly, describing and displaying a Whiteboard. A couple of the solutions addresses multiple problems however, ultimately, the different solutions must work together and be suitable to work on the target device.

3.1 Collaborative Editing

Collaborative editing of a document can be considered to be synchronous or asynchronous. Asynchronous collaborative editing corresponds to where only one editor actively edits a document at one point of time, while synchronous collaborative editing corresponds to when a multiple amount of editors are actively participating in the editing of one document, at the same time. In this report, only addresses synchronized collaboration, since it is the technique relevant to real time whiteboarding.

Concerning real time collaborative editing, some kind of synchronization between the participants need to be arranged, so that they can edit the same object at roughly the same time. Traditionally, two synchronization control approaches exist in reaching this behaviour [28], pessimistic concurrency control, referred in this report to as lock based, or optimistic concurrency control, referred to as lock free.

Collaborative editing between more than two participants require some kind of group communication to be exist. However, different collaboration strategies requires different styles of group communication guarantees. More on group communication and what can be accomplished can be found in Section 3.2.

3.1.1 Lock based

Lock based editing, or pessimistic concurrency control, relies on acquiring a lock before doing any editing on a given object. This assures that only one user can edit a given object at a time, which in some ways limits the parallel nature of the collaborative editing. At the time a user has locked an object, all other participants have to wait until the lock is released.

A typical scenario may look like this.

1. Acquire the lock on the object A, given that it is not already acquired.
2. Perform some edits to the object.
3. Release the lock on the object A.

Another relevant issue with lock based editing is when a participant actively editing an object crashes, leaving the object locked forever.

3.1.2 Lock free

Lock free editing, or optimistic concurrency control, allows participants to more freely edit an object in a more parallel nature. However, when doing this, there are risks that conflicts occurs, possibly making the versions of the objects differ between participants. Therefore, instead on a locking protocol, an algorithm for automatically resolving conflicts can be applied. This, however, may result in loss of data, when for example a users edits are overwritten during the conflict resolving.

3.2 Group Communication

The basic form of communication in networks is one-on-one communication where two entities are communicating with each other, and no other entity can participate. The problems related to this kind of communication, for example reliability and ordering, are addressed using different techniques. However, when it comes to communication with multiple entities, other problems occur.

If the application needs only to communicate with multiple parties without no reliability or order, it may simply use methods such as IP multicast which does not provide any kind of guarantee about transmission (see Section 3.3.2.1 for more information). For assuring any kind of reliability, such as order of packets or reliable transmission, more difficult problems occur.

In this section, three problems concerning group communication are discussed; membership, reliable communication and order. A number of research projects are also introduced. For design choices about group communication toolkit, see section 4.7.

3.2.1 Theory

This section describes problems concerning group communication, and methods for solving them.

3.2.1.1 Group Membership

Membership in a communication group refers to the problem when the clients in a group requires to know who are participating. A complete group communication system including membership management needs to include some kind of *service* [11] managing membership. A membership service could be implemented both as a centralized service or as distributed service to avoid a single point of failure disabling the whole system.

Such a membership service would have some main tasks: providing an **interface** for adding and removing clients, **crash detection** to keep track of failing clients, **notification** about changes to the group membership (adds, removes, crashes) and providing **group address alias** for clients to send messages which destinations are the current members of the group.

3.2.1.2 Broadcasts

Broadcasts in group communication refers to when messages are sent from one client to all other clients in the group. Broadcasts with different kind of guarantees are discussed further in this section.

Reliable Broadcasts When the transmission of broadcast messages has to be correct, an algorithm ensuring reliability has to be used. Such an algorithm has to ensure that either a broadcast message is received and processed intact at all receiving hosts, or none at all. There are various approaches to solve this, but they are not included in this report.

Total Ordering Total ordering within group communication refers to the guarantee that all broadcast messages are received in the same order, by all recipients.

A problem of ordering messages in a distributed system is to decide who exactly does the ordering. There are several ways of doing this and the most common algorithms can be classified as one of the following [9]: *fixed sequencer*, *moving sequencer*, *privilege-based*, *communication history* or *destination agreement*.

The role of a sequencer would be to be responsible of the ordering of messages. In order for a client to broadcast a message, it first has to ask the sequencer to order its message. The difference between a fixed sequencer algorithm and a moving sequencer algorithm is that the latter one allows the sequencer role to be moved between the group members, possibly by using some kind of election algorithm or similar.

In privilege-based and communication history algorithm, the sender is the one responsible for ordering messages. A privilege-based algorithm works in the way that a client can only send broadcasts when it has a certain token. This means only one client at a time can send a broadcast message, and a token is moving around between the clients in some way.

When total ordering is ensured using a communication history ordering algorithm, it is still the sender who is responsible for ordering, but a client can broadcast a message at any time, without waiting for a token. A broadcast message is normally sent containing a logical clock or physical time stamp. It is then up to the receiver to deliver the message to the application at the right time, usually by delaying the message until the order can be assured.

In the last mentioned algorithm class, destination agreement, ordering is done after receiving. For a client to send a broadcast message, it just sends it. Then later when the message is received, the receiving hosts agree on some kind of order. Ways of deciding this order can vary.

Atomic Broadcasts Broadcasting in group communication is to send a message to all the remote clients in the group. When the procedure of sending a broadcast message guarantees that messages are received intact and in the same order on all receiving ends, it is called a total order broadcast, or atomic broadcast.

3.2.1.3 Probabilistic Algorithms

An approach on guaranteeing some kind of reliability when transmitting messages with in a group, is by using a probabilistic algorithm [3]. This section shortly introduces a approach to probabilistic protocols for distributed systems based on gossiping called *pbcast*.

The *pbcast* protocols guarantees that either no or all correct processes will deliver a message, eventually. This property is called *eventual convergence*. However, this property may be too weka for many applications, however, this guarantees such as ordering and latency can be added [19]. These protocols also requires a static number of group members.

Probabilistic protocol can be designed to guarantee success of message transmissions to some degree, saying a message is *probably* going to arrive to all other ends, where the probability likely is very high. However, to guarantee that the probability of a successful message transmission is to a certain level, the system the messages is sent in must be within a number of factors. For guaranteeing that the probability that a message is successfully transmitted is f_a, a system must be have the properties that the probability of a unicast message transmission between two non-faulty hosts are being sent successfully is f_m and that the probability of a process failure is f_p. Both f_m and f_p are generally considered to be very small.

To perform a *pbcast* broadcast, the initiator sends an initial message to a random set of processes. When a process receives a message for the first time it it gossips the message to some other random set of processes. Each process only gossips once. A message also has a parameter *rounds*, specifying how many times a message may be

gossiped. The *rounds* parameter decreases every time a message is gossiped and when a process receives a message with the *rounds* value 0, the message gossiping stops.

3.2.2 Existing Research and Projects

3.2.2.1 Publications

A lot of research has been done on group communication by the Distributed Computing and System Research Group at Chalmers University of Technology. For instance research effort has been put into providing a layered architecture of services supporting multi-peer collaborative applications [14] as well as for achieving scalability for event-based peer-to-peer dissemination systems [13]. Deeper discussion about these research papers are out of scope of this thesis report.

3.2.2.2 Horus and Ensemble

In 1990, a research group at Cornell University began to work on a distributed group communication system called Horus [2]. The purpose of the Horus system was to solve the problems of the first-generation systems, which was among other complexity and lack of flexibility. The goal with this new system was to have a stable backend which one generic architectural framework where group communication interfaces are treated separately from their implementation. Using this design, one could use the Horus system with a group communication implementation matching the needs of an application.

The flexibility of the Horus system was done by using so called *micro-protocols*. One micro-protocol can be considered to be one layer in a protocol, where one layer is responsible of assuring one kind of guarantee. By stacking several micro-protocols on each other, one can have a system guaranteeing a larger set of guarantees.

An example of a micro-protocol is one that ensures message reliability by adding a sequence number to every message, retransmitting lost messages in response to Negative Acknowledgement messages (NAKs). This micro-protocol would have one part on the outgoing of messages adding a header containing the sequence number, as well as another part on the incoming of messages reading the header, maybe requesting retransmission or responding to retransmission requests.

In the Spring of 1996, the next generation of group communication systems, Ensemble, was developed. The purpose of Ensemble was to overcome the complexity of implementing the transformations done on the layered protocol. Thus, Ensemble was implemented using the O'Caml language[1]. By using theorem proving tools, most notably one called NuPRL, one could automatically transform the operations of the layered protocol stack of Ensemble, creating more optimized versions. In Horus, this would have to be done manually, without any automatic theorem proofs.

[1] O'Caml is a programming language unifying imperative, functional and object oriented programming.

Duo to Horus and Ensemble being very flexible toolkits, they have been used to implement high profile applications using both very large and small distributed systems.

Even though Ensemble is written in O'Caml, it is possibly to use the toolkit from various other languages for example Java, C and C++. However, running an application, the target platform has to be able to run programs written in the O'Caml language.

3.2.2.3 JGroups

JGroups is a general purpose open source/free toolkit written in Java for creating applications that uses group communication. It's highly dynamic and the developer can combine different techniques for specifying communication methods such as making use of multicast communication or simply sending unicast packets, as well as assuring various requirements such as group membership, broadcasts with total order or atomic broadcasts. JGroups is used in various small and large scale projects, such as JBoss[2].

Using JGroups is simple. The interface provides a `JChannel` class, which can be set up to suit the developers needs. Then, this channel is used to add message listeners and send messages. Messages sent to the channel will be sent using the algorithms specified in the instantiation phase. For a channel set up to use multicast for one-to-many communication, and to assure atomic broadcasts and total order, messages sent through it are guaranteed to have these qualities.

JGroups is based on a similar technique as of Horus and Ensemble, specifically the use of a layered protocol stack, as in the micro-protocols of Horus and Ensemble.

3.3 Data-link and Network

The data-link technologies for wireless local or private area network available on different Smartphones can vary slightly, where two methods dominate. The following section briefly summaries the features of the two major protocols.

3.3.1 Bluetooth

A common way for wireless short-range communication is Bluetooth[3], which is commonly available on many devices. It's decentralized in the way it does all communication peer to peer and doesn't require any central point to allocate addresses. However Bluetooth devices has a comparably short range and speed, compared to regular Wi-Fi implementations. Another drawback of Bluetooth is the inability to perform multicast communication which can be a setback in a distributed system based on multicast communication.

[2]JBoss is a open source/free Java based application server, with features such as clustering. Developed by Red Hat.

[3]Bluetooth is a wireless personal area computer network protocol originally developed at Ericsson.

3.3.2 Wi-Fi

Wi-Fi, or IEEE 802.11, is an other, widely adopted protocol for wireless communication. It's often referred to as Wireless Ethernet since it is used together with Ethernet and IP (Internet Protocol). However, in most cases it requires the use of a wireless access point[4]. This, however, is not a requirement since Wi-Fi can also be used in Ad-Hoc mode[5] thus enabling, in the same style as Bluetooth, direct point-to-point communication.

When participating in a Wi-Fi network using an access point it is common that the address allocation is done using a DHCP[6] however it is possible to allocate an address without such a service using Zeroconf, specifically the protocol Dynamic Configuration of IP Link-Local Addresses [5]. For more information about Zeroconf see section 3.4.1. In Ad-Hoc mode this is a more likely scenario, however, when this report was written, the support for decentralized automatic IP address allocation on devices was limited.

3.3.2.1 IP Multicast

Using access point based network, multicast communication is straight forward using multicast IP-packets, and even Ad-Hoc networks can perform multicast using several different protocols. A drawback of Wi-Fi networks is power consumption which compared to Bluetooth normally is a lot higher. This however is the price of having longer range and faster speed.

Note

The following sections are all based on using Wi-Fi as data link and network technology. For more detailed discussion about the design choices concerning data-link and network model, please read section 4.2.

3.4 Presence

A application running on an entity in a network can only do so much by itself. For the application described in this report, there is one major characteristics needed; configuration and setup should be automatic. For, without having any pre known information about who is available on the network, initiating communication with a remote entity a method for announcing presence as well as discovering remote presences has to be done. This section describes three methods for advertising a service as well as discovering it.

[4] A wireless access point works as central point of communication in a wireless network and enables all devices in range of the access point to communicate with each other.

[5] A Wireless Ad-Hoc network is decentralized wireless computer network, meaning it has no central point where all communication goes through, compared to a wireless network using an access point.

[6] Dynamic Host Configuration Protocol (DHCP) is a protocol for IP address allocation.

Among the different solutions, the Zero Configuration (section 3.4.1) part will be described with more detail, as it is the technology that is used later on. For further information of design choices concerning presence and discovery, see section 4.3.

3.4.1 Zero Configuration

In 1999 the IETF (Internet Engineering Task Force) set up the Zeroconf Workgroup for creating standards for simple fully automatic decentralized network configuration including network address allocation, host name translation, service announcing and discovery and multicast address allocation. The two techniques relevant to this section are host name translation and service announcing and discovery. For more information about Zeroconf network address allocation, see section 3.3.2.

3.4.1.1 Multicast DNS

mDNS (Multicast DNS [7]) describes a way making use of IP multicast communication together with regular DNS to discover and announce host names. However, as DNS can be used to more things than host name resolution, so can mDNS. In a mDNS system there are two general parts, a responder that advertises a DNS entry and listens for requests, and a client that requests DNS information from the network.

To advertise a host name using mDNS, the responder first sends out probe packets to the multicast address to assure that the desired name is not already taken. After the uniqueness is assured, the responder sends out announcements about the new DNS entry as DNS response packets to the multicast address.

3.4.1.2 DNS-Based Service Discovery

Using regular DNS packets to describe services and service information, one can make use of DNS-SD (DNS-Based Service Discovery [6]) which describes a way of using DNS packets to provide information about services. It uses DNS SRV [17] to provide an address and port number with a given service, and DNS TXT (Section 3.3.14 of Domain Names - Concepts and Facilities [26]) to provide a $key \rightarrow value$ list to the given service. The content of the TXT list is service-dependent, and used to provide more information except from address and port number.

Service names consists of three parts, instance name, service and domain, and is constructed in the following way: `<Instance>.<Service>.<Domain>`. Being not regular host names, service names are not restricted by the naming rules of host names and it is encouraged to use more describing names. For example, a hypothetical HTTP (Hyper Text Transfer Protocol) service with the name "Office coffee machine web interface" on the "foo.org" domain would have the complete service name `Office coffe machine web interface._http._tcp.foo.org`. If the mentioned coffee machine is available only

on the link-local network, the full service name would be `Office coffee machine web interface._http._tcp.local`.

3.4.1.3 Zeroconf based Service Discovery

It is possible to use DNS-SD for service discovery together with normal DNS-servers, however it is also, which is more important to this report, compatible with mDNS.

mDNS combined with DNS-SD can be used for discovering and announcing general presences, similar to the ones of Instant Messaging clients. A way of doing this is together with XMPP (Extensible Messaging and Presence Protocol [29]). For introduction of how XMPP works and how it can be used to address several problems associated with the problems described in this report, see section 4.3.

3.4.2 Simple Service Discovery Protocol

SSDP (Simple Service Discovery Protocol [15]), a service discovery protocol developed by Microsoft, is similar to the one discussed in section 3.4.1 as it uses a IP multicast to send announcements and discovery requests. However, SSDP service entries are pointers to HTTP (Hyper Text Transfer Protocol) services made available to announce their presence, and SSDP clients are HTTP clients enriched with the ability to discover services in the local network.

3.4.3 Service Location Protocol

SLP (Service Location Protocol [18]) is another protocol for announcing and discovering services in a local area network. It works similar to the other two discussed in this report by communication mostly using IP multicast. However, there are differences making this solution stronger in certain scenarios. In a SLP system there are three kinds of entities. There is the User Agent (UA) that is used by the client application to discover and initiate communication with a service. There is the Service Agent (SA) that advertises a service. And at last, not existing in the two others, there is the Directory Agent (DA) that collects service information and acts as a cache. The purpose of using DAs is to make the protocol scale better when more SAs and UAs are participating.

To discover services, a UA who knows nothing about the network sends a multicast request packet containing information about what kind of service the client is interested in. SAs then individually sends unicast replies to the UA. Even though DAs may be discovered using a decentralized manner the case when usaging them is irrelevant to this report as it is desired to assume no configuration is available.

SLP is used in several places, such as network printers and similar devices. It was supported by default in Mac OS X until 10.1 when they moved over to Zero Configuration (see section 3.4.1).

3.5 Free Software and Open Source Software

The term *Free Software* was originally defined by Richard Stallman in the mid eighties as he founded the Free Software Foundation to support and promote the free software movement. The Free Software Foundation is responsible for the development of the GNU Project[7], which Richard Stallman started in 1983 to develop a operating system. According to the Free Software Foundation a piece of software can be considered free, if the user of the software have the four following freedoms[8].

1. The freedom to run the program for any purpose.
2. The freedom to study how the program works, and change it to make it do what you wish.
3. The freedom to redistribute copies so you can help your neighbor.
4. The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits.

Open Source software is, however, generally referred to software which source code is provided under a license that meets the Open Source Definition, formulated by the *Open Source Initiative*[9]. In general, open source software is very similar, while open source more refers to the development model, and free software the social movement.

3.6 Platform

Depending on the design choices of the application, the platform alternatives gets limited. Choosing certain programming languages limits the choice of platform more than others, so making these choices has to be done while considering what platforms that is to be valued higher than others.

This section introduces a number of platforms that for this project was under considerations. Functionality is an important part of the decision making but other factors also play a big role. Licensing and cost plays some role, but also moral issues was be considered. On the market during the time this thesis work was conducted, there were several alternatives available, with different values, technology and philosophies behind them. This section gives a summary of some of them, in a technical point of view, as well as the other aspects.

[7] The GNU Project is a free software collaboration project. Software that is a part of The GNU Project are, among others: GCC - The GNU Compiler Collection, GNOME - The Official GNU graphical desktop environment, The GIMP - The GNU Image Manipulation Software.

[8] Taken from the Free Software Foundation website: `http://www.gnu.org/philosophy/free-sw.html`.

[9] The Open Source Initiative is an organization founded in 1998 dedicated to promote Open Source software.

Symbian OS

Symbian OS is a mobile phone platform developed most part by Symbian Ltd. but owned by the Symbian Foundation[10]. At the time this report was written, Symbian Ltd. was in the process of making Symbian open source, however, at that time, only some initial parts were made such, and only made available for Symbian Foundation members.

Symbian OS is the primary operating system on phones from several different manufacturers such as Nokia and Sony-Ericsson. During the time of this thesis project, it is the most commonly used Smartphone operating system available, however for a third-party developer there are a lot of complications. Concerning the development environment, Symbian OS provides a number of programming language interfaces for developing applications, among others C++[11], a port of the Python programming language and J2ME[12]. All these platforms provide interfaces to a subset of the devices functionality, often restricted to Bluetooth for WLAN[13], Multimedia features and general Internet functionality. However, for security reasons, Symbian OS limits installable programs to only a small set of functionality if the developer does not purchase a license. Strong sides of the Symbian OS is its event based design, making programmers able to write applications using very little power; the operating system can turn the CPU off when a program is waiting for an event.

Blackberry OS

Blackberry OS is another proprietary mobile phone platform created by the Canadian company Research In Motion (RIM) for their own set of devices. A third party developer can, except a set of tools for making browser based applications, make use of the MIDP[14] and CLDC[15] frameworks as well as RIM's own user interface framework. Similar to Symbian OS, a third-party developer has to purchase a license to make use of more advanced features on devices.

[10] Symbian Foundation is an non-profit organization founded to manage the open source development of the previously Symbian Ltd. owned operating system.

[11] C++ is a low-level object oriented C like programming language.

[12] Java 2 Platform, Micro Edition (J2ME) is a Java platform targeted for mobile devices. J2ME is much more limited than the standard Java edition.

[13] Wireless Local Area Network

[14] Mobile Information Device Profile (MIDP) is a specification of how to use the Java on embedded devices. MIDP, as J2ME is much more limited than the standard Java edition.

[15] Connected Limited Device Configuration (CLDC) is a specification of a Java framework targeted at limited mobile devices.

Android

Android is a open source/free mobile phone platform based on the Linux kernel as well as a set of open source/free applications and libraries. It was originally developed by Google Inc. but later taken over by the Open Handset Alliance[16].

Its goal differs from other platforms duo to its more open nature. Anyone with a computer and an Internet connection can download the complete Android platform and look, edit and redistribute the source code. The platform's development process is as well much more open and directly affects projects upstream[17]. Developers and users can file bug reports and follow them online, as well as edit the source code itself and submit patches.

Third-party developers have a much more powerful platform to work with, comparing to Symbian OS and Blackberry. One goal of the Android platform is to make the developer able to make use of as much of the phones functionality as possible, including standard built in functions such as multimedia player, Wi-Fi[18], built-in camera or any other part of the phones functionality. The development environment available is based on the open source/free Java platform Apache Harmony, developed by the Apache Software Foundation. Compared to J2ME, the Apache Harmony is a much more complete version of the standard Java edition, still it misses some of the packages.

It is also possible to write native code compiled to ARM code then accessed by the Java application.

iPhone OS

The iPhone OS is a proprietary operating system based on the Apple made Mac OS X made for the products iPhone and iPod Touch. Writing third party applications is done using the Software Development Kit which is only available to Mac OS X computers running on an Intel processor. However, iPhone OS provides the developer with quite a lot of possibilities, almost to the extent as Android does. Compared to for example Symbian or Blackberry, the developer is more free to use the features of the device, including graphics, Wi-Fi and other things. Different from the so far mentioned platforms, iPhone OS doesn't provide the programmer with a Java interface; instead it is only possible to use Objective C or C/C++.

[16] Open Handset Alliance is a business alliance of a number of firms including Google, Intel, HTC, Motorola and several other companies with the goal to develop open standards for mobile devices.

[17] Fixes in Android concerning libraries or applications used by Android is shared with the corresponding project.

[18] Wi-Fi is a trademark for products based on the IEEE 802.11 wireless communication standard.

Windows Mobile

Another proprietary operating system used in Smartphones is Windows Mobile, developed by Microsoft. It offers third-party developers to create applications using a provided Software Development Kit that works with Microsoft Visual Studio. One can write unmanaged code using Visual C++ or managed code by using the .NET Compact Framework[19]. The developer is made able to use features of the device such as Wi-Fi and graphics, is such features exist. However, using Windows Mobile one is restricted to a more close nature when it comes to protocols and toolkits.

Openmoko

Openmoko is a open source/free Smartphone operating system using Linux as kernel. It's based on various open source/free software such as GTK+ and QT, and uses X.org as its graphical environment, making it very similar to a normal Linux desktop computer. Development can be done using various programming languages such as C, Java and Python. The approach of Openmoko Linux makes it fairly easy to port existing Linux desktop applications, however since the computational power of a Smartphone device is much smaller than that of a desktop computer considered amount of work still has to be done to make a program suitable. A difference from the other mentioned platforms is that Openmoko has no consumer ready devices for sale, except for a beta phone meant for advanced users.

[19].NET Compact Framework is compact version of the .NET Framework created by Microsoft.

4 Analysis and Design

In this chapter the design choices of the application are discussed and motivated. It is divided up into the different parts which are all related to each other.

The designing process was divided up into a number of sub problems, that would be addressed one at a time. These sub problems were *presence in a network* and *one-on-one communication* and *collaboration* including group communication and synchronization of a whiteboard.

The goals (Chapter 2) of the application is the base of all the reasoning in this chapter.

4.1 Platform

The choice of platform has only so much to do with functionality. Other important aspects to consider is license issues, development platform restrictions and the cost of trying it out on real hardware.

The platform of choice became Android OS, introduced in Section 3.6, and the reasons for this are several. The following list argues why this decision makes sense.

Openness Android is built using open source/free software, and it is complete source code is available for anyone with an Internet connection to use. This is not only positive in a moral aspect, it may also in some cases increase the speed issues with the platform are addressed and fixed. The Android platform urges people to take part in the development by reporting bugs, submitting feature requests and supplying patches. During the time this thesis work was performed, two separate bugs, one involving multicast communication, one involving the 2D graphical toolkit supplied were addressed and fixed. Also, since there is no requirement to pay extra fees for using certain parts of the device hardware the cost of publishing an application is very low.

Powerful toolkit Developers writing programs for Andraid smartphones have many possibilities. This is made possible due to the extensive toolkit made for Android, making it possible to use almost any part of the phones functionality.

Cross-platform SDK The Software Development Kit made available that is used to write Android applications can be be used on Windows, Mac OS X and Linux,

however, by downloading the Android source code, one can compile a personal version and run in a virtual machine such as VirtualBox[1]. The Android SDK is made to work well together with Eclipse[2] but it is not a requirement.

Available devices One reason, which had a large impact on why Android became the target platform, was the availability of devices to test on. As employees around me were developing Android applications, there were devices for me to use during the development, with no extra costs required.

Since the primary language for developing applications on Android devices is Java, Java is the language used for implementation.

4.2 Data-Link and Network

The method of performing communication between devices was chosen to be Wi-Fi, introduced in Section 3.3.2. Reasons for choosing this instead of the other alternative, Bluetooth, are speed and features. Even though battery consumption will be higher, the fact that using Wi-Fi makes it possible to use multicast techniques and higher transmission rates makes it a more interesting for this research project. Fast responsiveness is especially important when dealing with graphical applications.

4.3 Presence using XMPP

For presence discovery and advertisement, the Zero Configuration model (see Section 3.4.1) is used. A major reason for this is the existence of the already defined protocol for general social presence and peer to peer communication: XMPP (Exstensible Messaging and Presence Protocol [29]).

4.3.1 Introduction to XMPP

XMPP is a presence, message and request/response protocol based on streaming bits of XML (Extensible Markup Language) over a TCP connection. The general and most common scenario is when a client connects to a server, authenticates, sends packets with given recipients as targets, the server delivers the packets to the correct recipient. Every entity in an XMPP network has a unique address of the form *user@host/resource*. The *user@host* part is similar to that of E-Mail addresses, specifying what user on what host, while the */resource* part specifies what entity. An entity in XMPP is a single XMPP session. There may be many entities using the same account, separated by the resource

[1] VirtualBox is an open source/free virtualization product developed by Sun Microsystems, Inc.
[2] Eclipse is a powerful open source/free software development environment.

name. A typical XMPP entity may be *alice@exampleserver.se/laptop* which could be Alice using her laptop to connect to her account *alice* on the server *exampleserver.se*.

An XMPP packet is an XML element which can be of three different types. All packets have three common used attributes; `to`, which specifies the recipient, `from`, which specifies the source and `id` which provides a unique packet id. The three basic packet types have three different purposes: presences, messages and inquiries.

Presences (`<presence/>`) are used to tell the server about the presence of the client, providing information such as personal message, presence mode (available, away, extended away, do not disturb and chat), and other information. These presences are then distributed by the server to all the users currently subscribed to the newly available user. A typical presence packet specifying the users mode to "Do not disturb" follows.

```
<presence>
  <show>dnd</show>
</presence>
```

Messages (`<message/>`) are used for sending asynchronous messages, such as normal chat messages as well as messages with other purposes. A normal chat message may look like the following.

```
<message to='romeo@forza' from='juliet@pronto'>
  <body>Art thou not Romeo, and a Montague?</body>
</message>
```

IQ (`<iq/>`) packets are used for synchronous communication, where one client requests data from or submits data to a remote entity. When performing inquiries it is required by the client to add a unique `id` value, since this value is to be used when replying, binding the response to the request. Another required attribute of IQ packets are the `type` attribute. There are four allowed values, two for inquiries and two for responses. The two available for inquiries are `get`, used for retrieving data, and `set` used for sending data. The other two are `result`, replied when a `get` or `set` are successfully processed, or `error` which is replied when the processing of the request failed. An example of a data request procedure with no errors follows (the dots (...) are used instead of real request data).

Listing 4.1: romeo@forza requesting data from juliet@pronto

```
<iq from='romeo@forza'
    to='juliet@pronto'
    id='get1'
    type='get'>
  ...
</iq>
```

Listing 4.2: juliet@pronto replies to romeo@forza request

```
<iq from='juliet@pronto'
    to='romeo@forza'
    id='get1'
    type='result'>
  ...
</iq>
```

However, as implied by its name, XMPP is extensible. Packets may be extended by adding sub elements with a given name and XML name space. An example is the Jingle extension, which provides functionality such as live audio and video transmission initiation, which simply is done by extending `<iq/>` packets with a `<jingle xmlns='urn:xmpp:jingle:1'/>` sub element. These extensions are defined and described in XMPP Extension Protocols (XEPs), however, XEPs may also describe extended functionality that doesn't just add extra elements to the basic XML elements. An extension interesting to this report is the Serverless Messaging extension (XEP-0174 [31]).

4.3.2 Serverless XMPP

What the server-less messaging extension does is it enables Instant Messaging like communication by defining a DNS-SD service type and describing how to use this to announce XMPP like presence information using mDNS (see Section 3.4.1 for introduction to the combination of mDNS and DNS-SD). Information such as availability (online, away or do not disturb), personal messages and information as well as Avatar (personal icon) are provided using DNS TXT records. The DNS-SD service type is called `_presence._tcp` and a link-local enabled instant message client would typically announce its presence as `user@host._presence._tcp.local` where `user` is the user name, and `host` is the host name of the computer the client is running on. The client also provides an address and a port number remote clients may connect to, and by connecting they can use End-to-End XML Streams (XEP-0246 [30]) to initiate an XMPP XML stream to communicate using two of the XMPP basic packet types; `<message/>` and `<iq/>`. Using presence packets would not make sense since presences are announced using mDNS/DNS-SD.

Serverless XMPP is already a popular protocol for local area instant messaging and is included in clients such as iChat on Mac OS X, as well as the cross-platform clients Gajim, Pidgin and Empathy.

Security considerations

Serverless messaging using XMPP doesn't by itself provide any form of security. Any user in the network can advertise any presence with any name, and the channel between them is usually unencrypted. To solve this, another XMPP extension exists that

specifies how to add a security layer to end-to-end streams, i.e. server-less streams, adding authentication and encryption. This extension, C2C Authentication Using TLS (XEP-0250 [25]), specifies how to make use of the TLS (Transport Layer Security [10]) specification, and describes three methods of adding security.

X.509 certificates For a normal usage this method is not convenient and too complex since all users need a key signed by third party certificate authority. However, in a enterprise scenario where certificates can be systematically distributed, clients can assure their authenticity having key pairs signed using the X.509 public key infrastructure [8].

OpenPGP A different approach of public key infrastructure security is by using the Web of Trust model described in OpenPGP [4]. Trusted OpenPGP keys can then be used during the TLS Handshake procedure to ensure a secure channel [24], while the keys them selves ensure authenticity. However, creating a Web of Trust is also a complex task and if no previous trust is set up between the users keys, no authentication or encryption can be assured.

Shared secret Another approach is to use a password (shared secret) that is exchanged prior to the initiation of the communication channel, and then use the SRP (Secure Remote Password [32]) extension to TLS, which uses the shared secret during the handshake procedure. This approach is easier to prepare than the two other, since no complicated key signing and sharing has to be performed, however, using public key infrastructure type of security may ensure stronger security, since shared secrets generally are shorter than encryption keys and may be forgotten or unintentionally shared.

Other considerations

As the target platform is a hand held device, and the network model is wireless communication, the amount of data transfered matters both to battery time and the amount of data can be transferred over time. For decreasing the amount of data sent over the communication channel, another extension to XMPP can be used to add compression to the stream [22], however according to tests run by Google, comparing battery life when using compression of an XMPP stream on a smartphone; their conclusion was that enabling compression actually decreases battery life since it took more CPU power to compress the stream than it saved in radio transmission power usage [27].

4.4 Extending XMPP

For being able to fulfill the goals of this application XMPP has to be extended. To do this, extension protocols have to be specified to cover the different problems. As clients

in a XMPP presence network may or may not have these extensions it is crucial to know who does and who does not, thus there has to be a way of telling a client's capabilities.

4.4.1 Client Capabilities

In XMPP, to know a client's capabilities and features, a technique specified in the extension Service Discovery [20] is used. It works in the way that a client who wants to know about a remote client's capabilities sends a special `<iq/>` request for requesting service information. The remote client then replies with information about available features and available services as well as client information, language and other information. A feature is represented as a unique string. For example, the feature string associated with remote controlling of clients is `http://jabber.org/protocol/commands`.

A client discovering a remote presence via server-less XMPP does not know much about the remote client's capabilities, and it would be inefficient to ask every single presence on the network if it supports the extension that is described in this section. To solve this another extension called Entity Capabilities [23] specifies a way of creating hash values out of the information that would be sent in a service information request response. In the normal XMPP scenario where a client connects to a server, this hash value is distributed as an packet extension to the `<presence/>` packet. When a client receives presence information containing an unknown entity capabilities hash value, it may request the service information associated with this hash value from the client, thus when it sees this particular hash value next time, the set of services available is already known without any extra communication.

Serverless XMPP also makes use of the Entity Capabilities extension by distributing the calculated values via the DNS TXT field also used for other presence information (see Section 4.3.2), thus lowers the amount of traffic used for discovering capabilities of remote clients.

To advertise the capability for a certain function described in an extension a feature string to the list that is replied in service discovery responses and the entity capability hash value is recalculated.

4.5 Spaces XMPP Extension

After knowing about each others presences, clients need a way to join collaboration sessions as well as start sessions and let other client join and collaborate. This section describes a new XMPP extension protocol that specifies how to request collaboration session advertisements as well as how to negotiate participation. In this report this extension is referred as *spaces*.

As when implementing any XMPP extension, it must not break compatibility with other clients implementing server-less XMPP in the network.

It should not depend on what medium is to be used for the group communication it self. It should enable the client to advertise collaboration sessions as well as provide a way of finding out what sessions a remote client is advertising. However, at some point there has to be a medium specific protocol, specified in a separate extension.

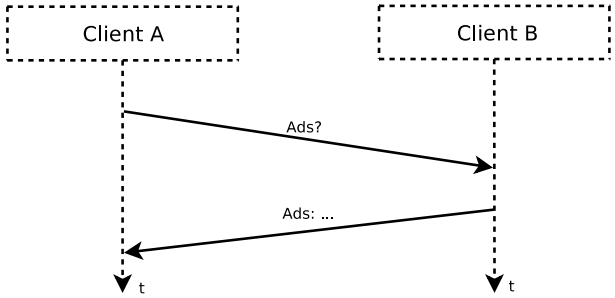
It is considered that, for joining a collaboration session, the client might need to provide additional data, such as pass phrase, encryption key or any other kind of data. This protocol should specify a general way of doing this.

4.5.1 Design

When a client enters an XMPP presence network, it needs a way to request information about what active collaboration sessions are available. To do this, without breaking backward compatibility, the protocol is designed so that a client can send a request to a remote client in the network asking for available advertisements, where an advertisement is some basic information about a collaboration space, including an unique identifier.

When a client wants to request a list of advertisements from a remote clientä it does so by sending an special IQ packet used for requesting space advertisements. The remote client may then reply with known spaces, as well as providing useful information. Every space is identified with a unique id, used later for negotiating participation. Figure 4.1 illustrates how a client (*Client A*) asks a remote client (*Client B*) for its advertisements. Detailed description of the usage of XMPP packets to do this follows in Section .

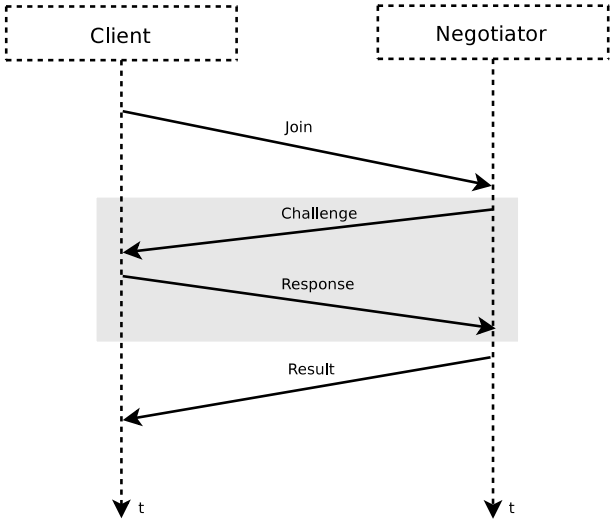
Figure 4.1: Requesting space advertisements



When a client is aware of the existence of collaboration space it wants to participate in, it may try to retrieve the information needed to start participating by negotiating participation with a client already active in the desired space, or a designated negotiator. To do this, a client sends a join request packet to the negotiator. If the negotiator decides to allow the client to join, it does so by providing information needed to join. The negotiator may also, before providing information, challenge the client. This is done by replying a join request with a challenge, for example a form to fill out. The client can then send another join request containing a challenge response.

Figure 4.2 illustrates how a client communicates with a negotiator about joining a space. The gray area marked in the figure emphasizes the procedure involving challenges.

Figure 4.2: Negotiating participation of a space



A complete specification of this extension is available in the appendix (see Appendix B). The rest of this section contains a description of the protocol specification and examples how it is used.

4.5.2 Protocol

Before initiating any of the following operations, a client must ensure that the remote client supports the protocol as well. To do this, it has to be made sure that the feature string `urn:itri:xmpp:space` is included in the remote client's service discovery response, as described in Section 4.4.1.

4.5.2.1 Advertisement

When a user want to know what sessions another client is advertising, a discovery query is sent. Such a request is bundled in the general XMPP `<iq/>` packet as a sub element named `spaces` with the XML name space `urn:itri:xmpp:space`. An example of a query packet requesting advertisements from a remote client named "juliet@pronto" is illustrated in Listing 4.3.

Listing 4.3: romeo@forza asks juliet@pronto for space advertisements

```
<iq type='get'
    to='juliet@pronto'
    from='romeo@forza'
```

```
    id='discover1'>
  <spaces xmlns='urn:itri:xmpp:space'/>
</iq>
```

A client receiving a request like this responds with a list of advertisements. An advertisement consist of only one required part, the UUID (Universal Unique Identifier) string. Other than the UUID, it may contain a description element where information such as name, short description text, content type and the number of participants can be supplied. The following example is a response to the request in Listing 4.3 where `juliet@forza` advertises two spaces. The first, a space containing a Scalable Vector Graphics (SVG) document named "Balcony Illustrations", and the latter, a session with no provided description.

Listing 4.4: juliet@pronto replies to romeo@forza's request

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='discover1'>
  <spaces xmlns='urn:itri:xmpp:space'>
    <ad uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'>
      <description>
        <name>Balcony Illustrations</name>
        <info>Clarification of intentions</info>
        <medium>jgroups</medium>
        <conten-type>image/xml+svg</content-type>
        <members>1</members>
      </description>
    </ad>
    <ad uuid='b00c4ee3-0232-48c1-977a-fa3982beab0f'/>
  </spaces>
</iq>
```

If no advertisements are available juliet@pronto would reply with an empty spaces-element.

4.5.2.2 Negotiation Participation

After using the procedure described in Section **??** a user might want to start editing one of the documents. This section describes how the Spaces extension specifies how to negotiate participation in a collaboration session with a remote client advertising it.

It is assumed that the client who are interested in participating in editing a document knows the associated UUID as well as a user advertising it. It may be taken for granted that the user has the ability to negotiate participation if the advertisement does not

contain a description element with a *negotiator* element specifying whom to negotiate with (for more detail description of this see Appendix B).

Worth noticing is that this part of the protocol does not specify how to start participating in collaboratively edited document, it only specifies how to request for information required to do so.

The first step a client does is to send a request to the remote negotiator. It does this by sending an IQ packet of the type *set* with a sub element `space` with the namespace `urn:itri:xmpp:space`. Required attributes are `state` - which while requesting has to be set to `join`, `role` - specifying if the client wishes to participate actively or as a spectator and `uuid` - specifying what space session the user wants to join.

Listing 4.5: romeo@forza asks juliet@pronto about a specific space

```
<iq type='set'
    to='juliet@pronto'
    from='romeo@forza'
    id='join1'>
  <space xmlns='urn:itri:xmpp:space'
         state='join'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'/>
</iq>
```

As stated in Section 4.5.1, the negotiator can challenge the joining client. This is done by replying with an identical `<space/>` element but the `state` attribute set to `challenge` as well as adding a sub element called `challenge` with a specific name space specifying what kind of challenge it is.

This extension defines a general purpose challenge type that uses the Data Forms extension [12]. The Data Forms extension specifies a way of including a form inside an XMPP packet. The form challenge type has the name space `urn:itri:xmpp:space:challenge:form`. A user receiving a form challenge must reply with a filled in form of the same type, as described in the Data Forms extension.

An example where *juliet@pronto* replies to *romeo@forza*'s request (see Listing 4.5) follows. She replies by asking him to fill out a form.

Listing 4.6: juliet@pronto replies to romeo@forza's request to participate in a session

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='join1'>
  <space xmlns='urn:itri:xmpp:space'
         state='challenge'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'>
```

```
    <challenge xmlns='urn:itri:xmpp:space:challenge:form'>
      <x xmlns='jabber:x:data'
         type='form'>
        <title>Capulet&apos;s orchard.</title>
        <instructions>My ears have not yet drunk a hundred words Of that
tongue&apos;s utterance, yet I know the sound:
Art thou not Romeo and a Montague?</instructions>
        <field type='text'
               label='Pass phrase'
               var='passphrase'/>
      </x>
    </challenge>
  </space>
</iq>
```

A client can be implemented in a way that forms bundled in a form challenge can be rendered automatically, however they can also have a pre defined structure as defined in the Field Standardization for Data Forms [21]. This extension also defines a Data Form field structure called `urn:itri:xmpp:space:challenge:form:pw` for simple password protection.

However, in the case that a client sends a challenge as in listing 4.6, this protocol extension does not specify how this form is to be filled in. In this case it is assumed that *juliet@pronto* asks a general question and wants *romeo@forza* to reply with an answer. To reply he may send the following IQ reply.

Listing 4.7: romeo@forza replies to juliet@pronto's challenge

```
<iq type='set'
    to='juliet@pronto'
    from='romeo@forza'
    id='join2'>
  <space xmlns='urn:itri:xmpp:space'
         state='join'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'>
    <response xmlns='urn:itri:xmpp:space:challenge:form'>
      <x xmlns='jabber:x:data'
         type='submit'>
        <field type='text' var='passphrase'>
          <value>Neither, fair saint, if either thee dislike.</value>
        </field>
      </x>
    </response>
  </space>
</iq>
```

Depending on if the challenge response is accepted or not, the negotiator may reply with either a IQ error reply, or a result containing a element named `medium` with a name space specifying what medium is used. What name space and how the medium element looks is not in the scope of this extension. The following example is of when *juliet@pronto* accepts the challenge response and replies with medium information (real medium information is not given since it is outside of the scope of this section).

Listing 4.8: juliet@pronto accepts romeo@forza's challenge response

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='join2'>
  <space xmlns='urn:itri:xmpp:space'
         state='result'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'/>
    <medium xmlns='...'>
      ...
    </medium>
  </space>
</iq>
```

4.5.3 Possible additions

A part from negotiation, this specification does only include functionality for making clients able to request advertisements only using IQ packets. This section mentions two ways to extend the functionality concerning discovery of sessions that are not yet covered.

Invitations

There already exists an extension for inviting a user to a multi user conference which works by sending a message packet containing an extra sub element used as an invitation. Invitation to a collaborative editing session could be done in a similar way by attaching a extra sub element to a message packet inviting a user as well as providing useful information needed for participation.

Announcing

This extension does not specify a way of announcing a new collaborative session in a server-less network. This could easily be done by defining new DNS TXT fields (see Section 4.3.2). However, this is a possible compatibility issue if the new fields are not

properly registered with the XMPP Standards Foundation, since they can collide with other clients using the same key names.

4.6 Collaborative Editing

This section introduces the way collaborative editing is performed.

4.6.1 Design

The problems of collaborative editing are several. If there are to be more than two parts, there is the problem of group communication. If the participants should have freedom to leave and then later continue, or if new participants are to have the ability to start participate in an already ongoing session there is the problem of sharing a document state. There is the problem of structuring the document. At last, there is the problem of synchronizing the state while still enabling the participants to edit the same document.

Goals mentioned related to the collaborative editing are the ability to reuse existing protocols. In XMPP, there are several, at the time this report was written, protocol proposals for performing both whiteboarding and more general collaborative editing. The protocol and algorithm used in this application is based on the previous work by Joonas Govenius [16] and Mats Bengtsson [1]. The places where the existing protocols differ compared with the one used in this application are described in this section.

This protocol specifies how to, between two or more people, collaboratively edit an XML document without using any locks. If more than two participants are to be able to collaborate, this protocol relies on some kind of group communication method that guarantees total order of messages, reliable transmission and group membership. A method of doing this in a server based manner is by using a multi user conference service to act as synchronization point. However, a goal of this this application is to use a decentralized distributed method for implementing group communication. More on the design of group communication may be found in Section 4.7.

4.6.1.1 Whiteboarding

As the method used for collaborative editing is based on that the document standard is compatible with XML. As a typical physical whiteboard mostly contains symbols and figures, as well as text, a document type need to cover all these parts. Another requirement related to the fact that it is supposed to be suitable to be run on a handheld smartphone is that it is preferable if the work space can be looked upon using different zoom states, without making the image quality any worse. In other words, it should be possible to look at a small part of the whole whiteboard without lower image quality. Using plain pixel-based protocols would for example not be good enough since zooming in on a particular part would result in that the image would be more bad the more

zoomed in the space is. A document type suitable for this is SVG (Scalable Vector Graphics), a XML-based specification of how to describe scalable vector graphics as an XML document. The version of the protocol used in this application is SVG Tiny 1.2.

4.6.2 Analysis

The algorithm used for keeping the document state synchronized between entities uses a lock free method making the entities automatically synchronize the state by using version numbers. This section describes how synchronization is performed and how the document parts are structured.

Document structure

As the document used in collaborative editing is an XML tree, every XML node is treated independently, and every XML node has an unique identifier and a version number. XML nodes includes elements (with name, name space and prefix), attributes (with name, prefix and character data), comments (with character data), texts (with character data) and processing instruction nodes (with target and data). All these nodes also has a weight indicating the order. A heavier weight makes the node appear later in the list, if the set contains more than one node. All nodes also has a parent value, which contains the identifier of the parent node (except the root node which has no parent). Consider the following table containing XML nodes and their properties.

Table 4.1: Example XML DOM internal structure

Type	Identifier	Parent	Properties
element	ID1	*N/A*	name = `html` namespace = `http://www.w3.org/1999/xhtml`
element	ID2	ID1	name = `head`
element	ID3	ID2	name = `meta`
attribute	ID4	ID3	name = `http-equiv` character data = `Content-Type`
attribute	ID5	ID3	name = `content` character data = `text/html; charset=UTF8`
element	ID6	ID2	name = `title`
text	ID7	ID6	character data = `Title goes here`

The weight and version values are omitted. In the protocol specification used, the default weight value is 1.0 and in the case when more than one node with the same

parent has identical weight values, the order is determined by which one was added first; first added comes higher up in the list. The version property, irrelevant to the rendering of the real XML DOM, is introduced later on.

Some basic rules exists of how the structure can be defined. For example only element nodes may have child nodes and names may not be empty strings.

The first row in table 4.6.2 contains an entry with identifier $ID1$ for the root element, since it has no parent. Looking at the parent column one can see that there is an element who are child element to the root element: $ID2$. Attributes follow the same pattern; node $ID4$ and $ID5$ both has $ID3$ as parent and are therefore attributes to the element $ID3$.

Generating the XML document using the data provided in table 4.6.2 would result in the following output.

Listing 4.9: XML DOM example (indentation added for clarification)

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Title goes here</title>
  </head>
</html>
```

4.6.2.1 Editing

For an entity in the collaboration session to create, edit or remove nodes it sends commands. There are three different commands, `new` - for creating new nodes, `remove` - for removing existing nodes and `set` - for changing the properties of existing nodes.

When creating a new node, the node gets an initial state. The new node must have a unique identifier, and it also has a initial version, default set to 1. When removing a node, all its records cease to exist. These two actions, create and remove, are non reversible. Concerning removing a node, it means commands that wishes to change the state of the node that was removed would be ignored. When adding a new node, there are no previous states, so it is trivial that it may not be undone (however, it may be removed using the command for removing a node).

The other command, `set`, is used for changing the state of an existing node. More description of how a state is changed and synchronized is found in Section 4.6.2.2.

As adding, changing or removing a single node in a tree may break the document structure, or create unwanted middle points, it may be useful to be able to group commands together. The protocol specifies a way of putting commands together making them handled one after another without any other commands interfering. For example changing the tree `<string lang='en'>Hello</string>` to `<string lang='sv'>Hej</string>` could be done by sending grouping one command editing the

`lang` attribute, and one for editing the text node into one action, avoiding any invalid middle point.

4.6.2.2 State synchronization

Every command performs a specific edit on a single node, and thus changes the state of that node. The following rules apply concerning commands are constructed and handled (for a mathematical model see Section 4.6.3).

1. A state contains a list of records with and their edit histories.
2. When a new edit command is constructed, the version of the command is set to $v + 1$ where v is the version of the state the edit command changes.
3. When sending an edit, it may immediately be applied locally to the sending client's own document tree.
4. When a new edit command is received, the state version is increased by one.
5. When a new edit command is received, if the target version of the command is identical to the new version of the state, the change is applied to the local tree.
6. When a new edit command is received, if the target version of the command is lower than the new version of the state, all the edits with a version equal or higher than the target version of the edit command is reverted and removed, thus undoing the changes they made to the node.

Considering that when more than one entity is concentrating on the editing of a single object in the tree, there is a higher probability of there being collisions. This makes this algorithm useful only when editing a document with many XML nodes because the probability that two entities are working on the same attribute is fairly low. However, if the document that would be edited is of a type which contains large groups of data, such as a text document, where it is more likely that many entities want to edit the same node (for example the same text node), this algorithm is not as suitable. However, since the target document type of this application is whiteboard, this is not an issue, instead it is a favorable variant of synchronization, since in whiteboards where the document structure is an SVG document, where editing is more spread out over the documents nodes.

The procedure can also be described using pseudo code, illustrated in Algorithm 4.1. The algorithm works in the way that it initially defines its state as empty, its version as 0. It then enters an eternal loop where it listens for new edit commands using the function $recvEdit()$ which returns the next edit. Another function seen is $targetVer(m)$ which takes a message as input and returns the target version. The $apply(m)$ and $undo(m)$

functions both takes edit commands as input and while $apply(m)$ applies the change described in the command, $undo(m)$ reverts the change to the way it was before the edit was applied.

$S \leftarrow \emptyset$
$v \leftarrow 0$
loop
  $m \leftarrow recvEdit()$
  $v \leftarrow v + 1$
  if $v = targetVer(m)$ **then**
    $apply(m)$
  else if $v > targetVer(m)$ **then**
    $U \leftarrow \emptyset$
    $S' \leftarrow \emptyset$
    for all $m' \in S$ **do**
      if $targetVer(m') \geq targetVer(m)$ **then**
        $U \leftarrow U \cup \{m'\}$
      else
        $S' \leftarrow S' \cup \{m'\}$
      end if
    end for
    while $U \neq \emptyset$ **do**
      $m'' \leftarrow highestTargetVer(U)$
      $undo(m'')$
      $U \leftarrow U \setminus \{m''\}$
    end while
    $S \leftarrow S'$
  end if
end loop

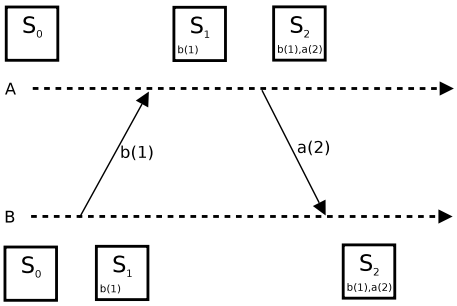
Algorithm 4.1: Processing of incoming edits

Figure 4.3 illustrates how two entities changes the state of an object. Explanation follows.

The dotted horizontal arrows shows the time. The squares with S_x indicates the current state, where x is the state version number. In the bottom part of the square the state is shown as a list of edits, for example $a(1)$ where 1 is the version of the edit command. The two clients involved are client A and B.

Figure 4.3 shows an initial empty state S_0 for both clients containing no edits and the version of their states are both 0. Then after some time B constructs and sends the edit $b(1)$. B casually applies its edit to its own document tree and when A receives

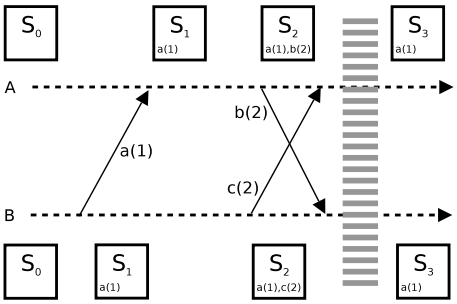
Figure 4.3: State synhronization



the edit, it increases itś state version and applies it to its own tree as the version of the edit corresponds to the new version of the node. Both A and B then has the state S_1, containing one edit: $b(1)$. Then after some more time A sends the edit $a(2)$. A can directly apply the edit to its own tree, while B can apply it when it receives the edit since the version equals the version of the current state. In the end the two clients contains two identical states, S_2 containing the two edits $b(1)$ and $a(2)$.

Figure 4.4 illustrates when two clients sends edit commands at the same time, and how the state automatically synchronizes ending up identical.

Figure 4.4: State synhronization



Initially, client A and B both contains the empty state S_0. At some time B, as in the previous example, sends an edit command $a(1)$ making A and B both have identical states S_1 containing the edit $a(1)$. However, after this, both A and B sends their own edits $b(2)$ and $c(2)$ at the same time. Both casually applies their own edits to their document trees, ending up with two different states; A's state S_1 contains the two edits $a(1)$ and $b(2)$ while B's state S_1 contains $a(1)$ and $c(2)$. Later, when the edit commands have arrived at the remote destination, the synchronization takes place, shown as the grey lines in the figure.

The synchronization works in the way that when an edit command with a version v is received, if $v < s + 1$ where s is the version of the state the receiving had when the

edit was received, all edits with version v or higher are reversed and removed, making the state go back in history. As shown in figure 4.4, when A receives the edit $c(2)$ it compares the version of the edit to its own states version. Since A's S_2 version has the value 2, and $2 < 2+1$, it reverts its state, removing edits with version 2 or higher, ending up with an state with the stat version increased by one, S_3, but containing only the edit $a(1)$. The same happens to B when receiving the edit $b(2)$. Finally, both A and B ends up with two identical states containing only the edits performed before the collision.

Figure 4.5: State synhronization

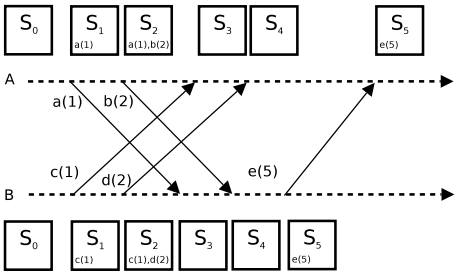
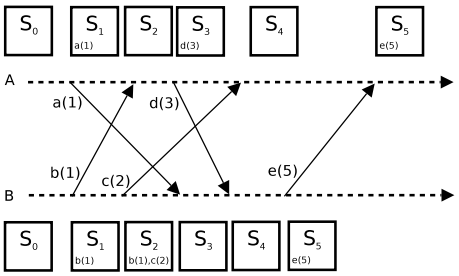


Figure 4.5 illustrates another example where a state is reverted to a previous state. As seen when A, while having the state S_2 consisting of the edits $a(1)$ and $b(2)$ receives the edit $c(1)$ it removes the edits with version 0 or higher, increasing its state version by one, ending up with an empty S_3. The second received edit $d(2)$ only increases the state version to 4, not making any changes to the set of edits. As illustrated, after some time B decides to send another edit, $e(5)$ which A later receives. As the version of A's state is identical to the version of the edit, it applies the edit. In the end (right part) of the figure both clients has the state S_5 containing the edit $e(4)$.

Another figure (4.6) illustrates how two states synchronizes using state version numbers.

Figure 4.6: State synhronization



Correlation between nodes

A node in a XML document is in some way always related to some other node or nodes, except the case when only one root element exists. For example attributes are always a child node of an element, so attributes and elements may be related. This means, that when creating, removing or changing the state of a node, it may affect some other node. Consider, for example, the case when removing an element. If the element that is to be removed has a number of sub nodes, such as sub elements or attributes, these are also effected: when a parent of a node is removed, that node is removed as well, as well as all its sub nodes.

Another example is changing the parent property of a node. Changing the parent property of an attribute, in other words moving it, affects two elements: the old parent and the new parent.

4.6.3 Correctness

This section tries to argue that the states of two different clients always are identical when all edits in motion has been handled.

Introduction

Consider that all the states, which are sets of edits e_m, where m is the edit's target version (referred to as edit version, edit's target version or target version). When an edit e_m is received, a state change is performed. The new state always increases its version number by 1. The state change is shown as the function S_{n+1}, taking the edit e_m as input. S_n is the state when e_m was received and n is the state version when e_m was received.

$$S_{n+1}(e_m) = \begin{cases} S_n \cup \{e_m\} & \text{if } m = n+1 \\ \{t_v : t_v \in S_n \wedge v < m\} & \text{if } m \leq n \\ S_n & \text{if } m > n+1 \end{cases} \tag{4.1}$$

It is trivial to see that for every edit which target version is identical to the version of the receiving nodes state, the state grows. However, to insist correctness, it is needed to be shown that when an edit message crosses any number of other edit messages, both the hosts state will revert back to the state before the first message that was crossed.

An edit crossing another edit refers to when, before a host A receives an edit from host B, A sends its own edit, unaware of the changes it is going to receive will cause a conflict. See Figure 4.4 for graphical representation of two edits crossing.

In other words, what is needed to be shown is that when an edit message crosses another edit message, the condition $m \leq n$ will be true, and that any edits that are sent

after the first edit message that was crossed, but before the edit that was crossed was received, will be ignored ($m > n + 1$).

For doing this we first have to clarify how the state changes when sending an edit. Consider that the current state is T_n where n is the version. A new edit message e_m is created, applied to the state using the function T_{n+1}, where $n + 1$ will be the new state version, taking the edit message as input.

$$m = n + 1 \tag{4.2}$$

$$T_{n+1}(e_m) = T_n \cup \{e_m\} \tag{4.3}$$

Part I

Here it is shown that an edit crossing another edit, the two hosts will result with equal states and state versions.

Initially, the two hosts A and B have their own initial empty states A_0 and B_0.

$$\begin{aligned} A_0 &= \emptyset \\ B_0 &= \emptyset \end{aligned} \tag{4.4}$$

A sends an edit a_1 and B simultaneously sends an edit b_1. Immediately after the edits has been sent, before any has been received by their remote hosts, their corresponding states will, given the equation in 4.3, result in the transitions shown in 4.5.

$$\begin{aligned} A_1 &= \{a_1\} \\ B_1 &= \{b_1\} \end{aligned} \tag{4.5}$$

What is to be shown is that when A receives B's edit b_1, A's state will revert; as $1 \leq 1 + 1$ thus edits in A_1 with index equal or greater than 1 will be dropped.

According to the formula in Equation 4.1, the result of the transition is shown in 4.6.

$$A_{1+1}(b_1) = \emptyset \tag{4.6}$$

The same of course happens to B's state when edit a_1 is received, as shown in 4.7.

$$B_{1+1}(a_1) = \emptyset \tag{4.7}$$

Part II

Now consider that B sent a number of edits before receiving the a_1 from A. What is needed to be shown is that all edits sent by B that was sent before a_1 was received will be discarded by A. The case when B sent 0 messages is trivial. The base case, if B sent 1 message c_2, the state set will not grow as presented in equation 4.1 ($n \leq m$, $m = 2, n = 2$) and shown in Equation 4.8.

$$B_{2+1}(c_2) = \emptyset \tag{4.8}$$

Left to show is that if e_{m+1} is received it will not make the state set grow. First lets assume that $p = m + 1$ making the edit e_p. In the conditions of equation 4.1 the three conditions are $p - 1 = n + 1$, $p - 1 \leq n$ and $p - 1 > n + 1$. This can be expanded to $m = n + 1$, $m \leq n$ and $m > n + 1$. When A receives e_{m+1}, if A during this time does not send any edits, the state version of A will always be $m + 1$. Using this property when expanding the mentioned equations, we get that $m \leq m+1$, the second condition, which does not make the state set grow.

Part III

It is also needed to show that when all crossing edits has been received on both sides, their state versions are equal. This is fairly trivial since every sent edit increases the state version by one, and every received edit increases the state version by one. Thus, starting on version 0, if A sends x edits, it will increase its state version to x. If B sends y edits, when A have received all of them A's state version will be $x + y$. For B, after sending y edits its state version will be y; and after receiving A's x edits, its state version will be increased to $y + x$. And since $x + y = y + x$ it is shown that A's and B's state versions will be equal after all crossing messages are processed. That no messages are lost are assured by the reliable atomic breadcast protocol used in the group communication.

Part IV

This part tries to shows that when A sends a number of edit while B sends none, after all messages has been processed both A and B will end up with the same state set.

In this part, the arrow symbol ($\rightarrow$) is used to illustrate a state transition. For instance, consider we have the state $S_0 = \emptyset$. A transition for when processing the message e_1 is written $S_0 \xrightarrow{S_{0+1}(e_1)} S_1$ meaning that S_0 transitions into S_1 either using 4.1 or 4.3, depending on if S is the sending entity, or the receiving.

As always, before any messages has been transmitted, the state of all nodes begin as empty sets.

$$B_0 = \emptyset$$
$$A_0 = \emptyset \tag{4.9}$$

To show the correctness of the stated condition induction will be used. For doing this, we first define some statements: $A'(n)$ and $B'(n)$. $A'(n)$ means *n transitions of the state A given n messages from A*. The meaning of $B'(n)$ is the same except that it is B that is undergoing the transitions. The transition equation used depends on if the node is the receiver or the sender. In this case, as stated, A is the sender, thus using Equation 4.3 and B is the receiver, using Equation 4.1.

Equation 4.10 shows how $A'(n)$ and $B'(n)$ are defined. Considering that these statements are transitions, it means that the resulting value (state set) is that of the result from the last transition.

$$\begin{aligned} A'(n) &= A_0 \xrightarrow{A_{0+1}(a_1)} A_1 \xrightarrow{A_{1+1}(a_2)} ... \xrightarrow{A_{(n-1)+1}(a_n)} A_n \\ B'(n) &= B_0 \xrightarrow{B_{0+1}(a_1)} B_1 \xrightarrow{B_{1+1}(a_2)} ... \xrightarrow{B_{(n-1)+1}(a_n)} B_n \end{aligned} \tag{4.10}$$

The induction hypothesis $P(n)$ is shown Equation 4.11. It can be read as *After n transitions given n edits from A, both A and B will result in identical states.*

$$A'(n) = B'(n) \tag{4.11}$$

The base step, $P(1)$, is when A sends only one message. As shown in Equation 4.12 for the left hand side of $P(n)$ and in 4.13 for the right hand side, both sides ends up with two identical states, containing only a_1.

$$A'(1) = A_0 \xrightarrow{A_{0+1}(a_1)} A_1 = A_0 \cup \{a_1\} = \emptyset \cup \{a_1\} = \{a_1\} \tag{4.12}$$

$$B'(1) = B_0 \xrightarrow{B_{0+1}(a_1)} B_1 = B_0 \cup \{a_1\} = \emptyset \cup \{a_1\} = \{a_1\} \tag{4.13}$$

In the induction step, $P(n+1)$, we need to show that $A'(n+1) = B'(n+1)$ holds. Equation 4.14 shows how the left hand side can be transformed, while 4.15 shows the right hand side. One can see that the first n transitions of $A'(n+1)$ are identical to the ones in $A'(n)$. The same conclusion can be drawn for $B'(n+1)$ but for $B'(n)$. Considering that the left hand side results in $A_n \cup \{a_n\}$ and the right hand side in $B_n \cup \{a_n\}$, since in the base case shows that $A_n = B_n$ holds for $n = 1$, one can conclude that by induction, $P(n)$ holds for all n.

$$A'(n+1) = A_0 \xrightarrow{A_{0+1}(a_1)} A_1 \xrightarrow{A_{1+1}(a_2)} ... \xrightarrow{A_{(n-1)+1}(a_n)} A_n \xrightarrow{A_{n+1}(a_{n+1})} A_{n+1} \tag{4.14}$$

$$= A'(n) \xrightarrow{A_{n+1}(a_{n+1})} A_{n+1} = A_n \cup \{a_{n+1}\}$$

$$B'(n+1) = B_0 \xrightarrow{B_{0+1}(a_1)} B_1 \xrightarrow{B_{1+1}(a_2)} ... \xrightarrow{B_{(n-1)+1}(a_n)} B_n \xrightarrow{B_{n+1}(a_{n+1})} B_{n+1} \tag{4.15}$$

$$= B'(n) \xrightarrow{B_{n+1}(a_{n+1})} B_{n+1} = B_n \cup \{a_{n+1}\}$$

4.7 Group Communication

The XMPP scope does not contain decentralized group communication protocols, but for solving the problems of this application, a way of communicating similar to a multi user chat[3], except not being dependent on a central service, is needed. Regular server-based multi user chats contain group communication qualities such as total order and reliable transmission. Thus methods and specifications for these methods incorporation with XMPP has to be created.

The protocol toolkit used in the application of this report is JGroups (see Section 3.2.2.3 for an introduction to JGroups), since it provides a highly flexible toolkit suitable for both large and small scale communication. Also, as it is written completely in Java it is able to run on the Android platform with minimal modifications. For more about the implementation see Section 5.4.2.

What the JGroups toolkit provides is a packet based communication model called channels where it is possible to send Java objects (that are serializable) or simply raw byte arrays. This section describes how JGroups is used within this application. A part of the application is the extension protocol documentation, specifying the usage of JGroups toolkit in this application. The protocol extension is available as Appendix C.

4.7.1 Group Communication Techniques

As JGroups is a highly flexible toolkit its usage has to be specified to suit the needs of this application. The requirements that has to be fulfilled are reliable transmission, total order of broadcast messages and group membership. Protocols for these problems are included in JGroups from the beginning, and protocols based on probabilistic algorithms exist (see Section 3.2.1.3 for more information).

To provide a more detailed documentation of the usage of JGroups a complete description of the properties used in JGroups may be found in Appendix .

[3]A multi user chat (MUC) in XMPP is a service based conference system where users can join a *room* where messages will be distributed too all participants.

4.7.2 Bridge to XMPP

Since JGroups is not related to XMPP, which is the protocol is used to discover presences in the network, some kind of bridging between them needs to be done. In Section 4.5 an extension specifying a way of discovering, advertising and joining of collaborative work space sessions are described. This section explains how the extension in Section 4.5 is extended with the functionality to discover, advertise and request information for joining JGroups sessions.

To do this, a type of `<medium/>` element providing necessary information needed to initiate a JGroups channel as well as starting collaboration is defined. The protocol specification specifies a basic set of default properties related to initiating a JGroups channel, such as IP multicast address, multicast port and a set of protocols to use for the various group communication techniques.

The usage of channels are designed in the way that every group session has its own channel, and every channel has its own unique identifier. This means that the basic case, when no special channel properties is specified, the only thing that is needed for a client to start participate is the unique id of the channel[4].

An example follows of a possible packet responding to a join request, containing a `<medium/>` element providing JGroups specific information.

Listing 4.10: juliet@pronto replies to romeo@forza's join request providing information about a JGroups channel

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='join2'>
  <space xmlns='urn:itri:xmpp:space'
         state='result'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'/>
    <medium xmlns='urn:itri:xmpp:space:medium:jgroups'>
      <cluster>juliet@pronto:f13f9f26-cce9-4171-80d8-e251c67bb0a1</cluster>
    </medium>
  </space>
</iq>
```

4.7.3 JGroups Message Protocol

Since one of the design goals of the protocols of the application is to make every part reusable in one or another way, it is preferred to use a way of communication within

[4]The version of the protocol extension of the time this report was written does not specify how to include any extra parameters, however it is simple to add parameters for example for multicast addresses and ports.

a JGroups channel that can be to some degree reusable in other places. With this in mind, a possible choice is to use a XMPP-based protocol for the content of the channel messages. This can be beneficial in different ways, for example there is no need to implement a new parsing mechanism that is only to be used for the JGroups message part. Another reason is by using XMPP-based packets the protocol in JGroups is a fully extensible in the same way that XMPP is, it is already supporting synchronous and asynchronous communication techniques, and many existing protocol extensions can be used, for example for formatting of messages using XHTML and other.

In the case of this application, the design choice made is a JGroups message contains one or many XMPP packets, which are encoded as raw UTF-8[5] strings.

Presences have different properties. As JGroups manages group membership, presences are limited to what ever information is provided, which is IP address and port number. The presence information discovered using XMPP can not be linked to any of the members of a JGroups channel without having to define a protocol for doing just this. Since it may be inconvenient for people communicating in a social manner to address each other using IP addresses, a message can be sent containing a nick name, possibly using the User Nickname XMPP extension.

When receiving messages using the JGroups interfaces, the programmer knows the source and if the message is a unicast message or a multicast message, thus the usage of the `to` and `from` attributes of XMPP packets are not needed and not even a reliable source of information. If the message is a unicast message it is assumed that the message is meant for the receiving entity, and if the message is a multicast message it is assumed that the message is a group chat message.

[5] A character encoding compatible with ASCII which supports multi-byte characters including characters from many different languages.

5 Implementation

This chapter addresses the design choices, decision making and issues encountered concerning the implementation of application. How various parts and toolkits are made to work together are explained. Issues encountered during the implementation are addressed as well.

5.1 Platform

As the target platform is Android the program is to be written in the Java programming language. Many parts of the application as possible is to be written with portability in mind, meaning that as many parts should be usable on any platform using the Java programming language. The user interface is the part only usable on the Android platform.

The implementation of Java used in Android is Apache Harmony, from the Apache Software Foundation, and by the time this work was done, the version available on the Android platform was not a complete version of the Java 2 Standard Edition, thus some portability issues of existing software could occur. Discussing about this is available in Section 5.5.

5.2 Presence

Presence and one-on-one communication was decided to use the Serverless XMPP method. A goal was to use and extending existing free/open source software. A library widely used and considered stable and secure is the XMPP library Smack, developed by Jive Software. Smack is a flexible, feature rich[1] and extensible. What is missing for this application is server less XMPP.

5.2.1 Serverless XMPP

Implementing server less XMPP in Smack can be divided up into two major parts; an mDNS/DNS-SD server and client for presence in a network, and the one-on-one communication done via XML streams. This section gives a basic description of how

[1] Smack includes features such as File transfer and Audio chat (Jingle).

Smack is designed and how parts of Smack is reused to implement server less XMPP into Smack.

Network Presence

There were two alternatives concerning using mDNS/DNS-SD implementations. One, using the Avahi daemon, the default on many Linux distributions, which by using DBus[2] can be accessed from any programming language supporting DBus, including Java. However, since it requires a daemon running in the background, usage on Android may be complicated.

The other approach was using the JmDNS toolkit, which implements basic support for mDNS/DNS-SD in pure Java. For using this program, some missing features had to be added: subscribing to information field updates from services, and updating information fields of advertised services.

The JmDNS source tree was forked[3] and the needed features were implemented in a similar way the existing functionality is implemented.

The JmDNS implementation is structured in the way that a service may have a current task running. For example initially, the task is to advertise the service. To implement updating information fields a new task had to be added, which only advertises the TXT fields of the DNS-SD record without unregistering and registering the service again. This is needed to support the ability to update information such as changing of status message or mode.

The other addition added was the ability to subscribe to updates from remote presences.

One-on-one Communication

The mDNS/DNS-SD service record contains an IP address and a port number. After having discovered a remote presence, connecting to the address and port given in the record using TCP, the protocol used is fairly similar to the one used in regular XMPP. For using this stream to send XMPP messages, as well as processing incoming messages, the parsing mechanism used in Smack could be reused.

Refactoring The Smack library contains set of classes used for parsing and processing XMPP packets. Generally, they can be divided up into *PacketReader* - used for parsing the XML stream creating XMPP packets, *PacketWriter* - used for transmitting XMPP packets, *XMPPConnection* - used for creating a new XMPP sessions and processing incoming packets parsed by PacketReader. The parsing of packets are done by using packet providers. What a provider does is, for a given XML name space and element

[2]DBus is a inter-process communication method.
[3]Git repository available at http://github.com/jadahl/jmdns

name, it constructs a specific packet extension. By implementing a new packet extension class, and by implementing providers for constructing new packets given an XML stream, one can extend the parsing mechanism to contain new packet types. Extended functionality and interfaces for such is made available, often by adding a listener listening for new created connections. This way, when a user initiates a XMPP session by creating a XMPPConnection instance, all extensions that are activated detects a new instance and adds its features to the connection, while implementing an interface using documented classes.

To add support for server less XMPP still using theses features, Smack had to be refactored in some ways.

Figure 5.1 shows a class diagram of how the version of Smack that this implementation was based of is structured. Figure 5.2 shows a class diagram after the refactoring was done.

Figure 5.1: Original Smack Design

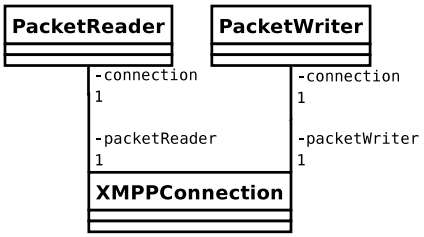
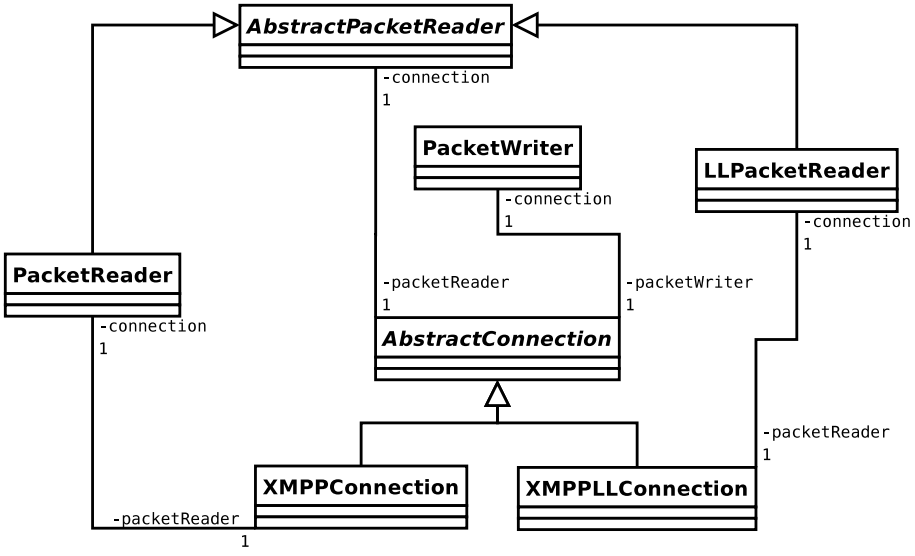


Figure 5.2: Refactored Smack Design

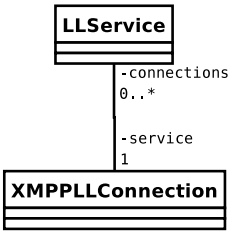


The goal is to reuse as many parts of the original Smacks functionality as possible,

and this was done by reusing parts of the packet reader and writer as well as the XMPP connection class. The functionality of the PacketReader and XMPPConnection classes that could be usable by server less XMPP was moved to a more general class, *AbstractPacketReader* and *AbstractConnection*. The functionality specific to server based XMPP was kept in XMPPConnection which also inherits the functionality of AbstractConnection. PacketReader was refactored in a similar way. Other parts of Smack was altered to, in the degree possible, make use of AbstractConnection instead of XMPPConnection, when the functionality was not based on that it is a server based connection. Examples of using the more multi purpose connection class instead of the server based follows later in this section. Another kind of connection class was then created, *XMPPLLConnection*, which has the functionality of a temporary one-on-one server less XMPP connection.

A server less XMPP connection has some major differences from server based. In the normal case, as long as the session is active, the connection is active. However, when dealing with server less XMPP, the connection is only active while communication is performed, and when not used for a while, it is terminated. Another major difference is that in normal XMPP there is one connection only, and messages to remote clients are sent through this connection, while in server less XMPP there is one connection per client. The advertising of presence differs as well. In server based, this is done by sending a presence packet, while in server less mode done by advertising a mDNS/DNS-SD service. While the Java class XMPPConnection is used as an interface for a XMPP session, this style is not doable using the server less XMPP connection class using XMPPLLConnection, thus an separate interface for server less XMPP sessions is needed; the Java class *LLService*. This is arguable beneficial since it would take care of creating new connections, sending messages through them, as well as terminate them when not used. It could also act as an interface for the mDNS/DNS-SD service. Figure 5.3 shows the relation between LLService and XMPPLLConnection.

Figure 5.3: LLService relationship with XMPPLLConnection



As the functionality and naming of the refactored XMPPConnection is the same as before the changes all features implemented using the previously mentioned self adding methods will still work as before, with no changes needed to be made. The refactoring keeps the library completely backward compatible.

Reusing extra functionality However, it is preferred to port some of the extended functionality to be usable with the server less sessions. One extended functionality already implemented for server based XMPP needed to ported to be usable also using server less: Service Discovery (introduced in Section 4.4.1). A simple approach was made by simply make the service discovery class *ServiceDiscoveryManager* use AbstractConnection instead of XMPPConnection, though still listening only for new XMPPConnection's (A connection listener added in the same way as before any changes still only gets notified about new server based connection). As ServiceDiscoveryManager uses a per connection class interface, another class, LLServiceDiscoveryManager, was implemented to map the functionality, creating an interface for using service discovery functions together with LLService.

Speeding up Service Discovery As mentioned in Section 4.4.1, the use of service discovery can be speeded up by using entity capabilities. Entity capabilities was implemented from scratch to be used both with the general service discovery manager as well as the link-local one. The implementation part that differs between these two use cases is the way the information is retrieved and distributed. In the link-local case, the entity capabilities data is provided via the TXT fields of the DNS-SD service entry, while in the general case the information is attached to presence packets.

5.3 Client Application

The rest of the protocol extensions described in this report are not extensions that, at the time this report was written, was submitted to the XMPP Standards Foundation. It was therefore not clear in what license the source code and documentation would be published in. Also, as these protocol extensions are mainly devoted to the purpose of this application, it was a logical step to create a new project tree.

One goal of this new project tree was to create a client library, designed to make the implementation of a user interface application easy. It was designed to act like a interface generalizing the use of the two different session types in Smack; server-based and server-less. Even though only the server-less part was implemented, the library was designed this way to make it simply and non-intrusive to add support for server-based. Implementing this kind of interface is required since the difference between using the interfaces for server-based and server-less communication.

While this library interface makes it more simple usage of these two XMPP session types, because of the differences in usage, there are still some difference that needs to be taken care of by the user interface, for example the roster (contact list) and available feature set. The functions that are identical to the two methods are sending messages and get/set queries, enabling and disabling, setting status and adding packet listeners.

Adding functionality to this client library is done in a similar way as when adding functionality to the Smack library. One creates an interface that is local to a session. This interface is implemented so that it automatically activates itself by adding static session listener and packet listeners.

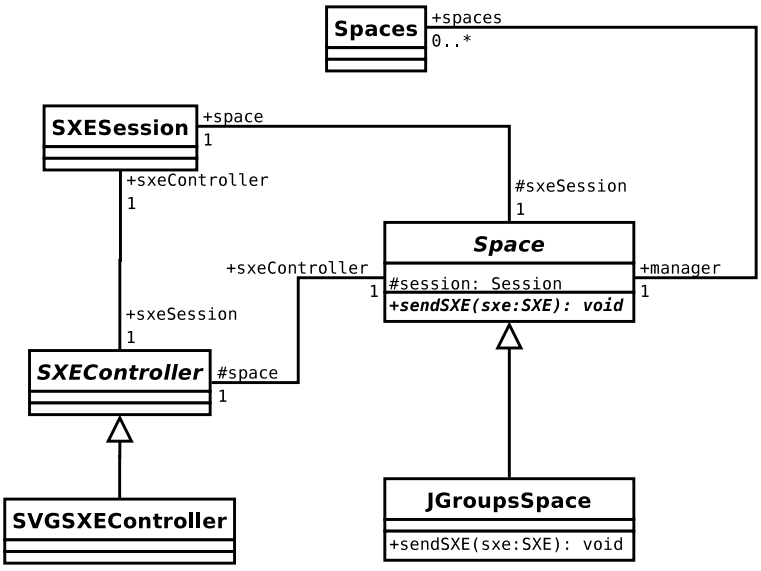
5.4 Spaces

The protocols introduced in Section 4.5, Section 4.7 and Section 4.6 is implemented as extended functionality in the client library. These three parts are closely related, and referred as *spaces* or a *space*. Spaces consists of several parts, all described separately in this section. The subjects brought up are discovery and advertisement, a transport medium (the introduced space medium in this report is the group communication toolkit JGroups) and the collaboration protocol.

Each space session is implemented as an instance of an abstract class `Space` implementing functions unrelated to medium and document type, as well as creating a general interface by having a set of abstract functions. The abstract functions not implemented by the `Space` class are the functions related to communication functions, thus, to instantiate a `Space` instance, one instantiates a medium specific space. A medium specific space class is introduced in Section 5.4.2.

Figure 5.4 shows a UML diagram of how the relationship between the different parts of the spaces extension. The various parts are explained in more detail throughout this section.

Figure 5.4: Spaces Extension



5.4.1 Advertisement and Discovery

The protocol described in Section 4.5 is implemented as an extended functionality to the client library described in Section 5.3. As the implementation described in this report only contained server less sessions, it is only possible to discover and advertise locally, but the design is done in the way that it only uses functionality common for both session types. Adding support for server based communication in the client library would transparently add support for server based advertising and discovering, in the way done in the extension protocol described in Section 4.5.

5.4.2 Group Communication

The design goals of the application was to create a work space shared between any number of users, with no central point of failure. This led to the decision to use the JGroups toolkit to implement a group communication medium for collaborative editing. This is implemented as a class inheriting the `Space` class adding communication methods implemented to use JGroups. Figure 5.4 shows an UML diagram illustrating the relationship between the abstract space class and the JGroups specific space class.

5.4.3 Collaborative Editing

Collaborative editing consists, implementation wise, of two major parts. One part is the processing of incoming commands that makes changes to the local state. The other part is the construction of new edit commands that is to be transmitted over the communication medium.

The processing is implemented in the non-abstract class `SXESession` (SXE stands for Shared XML Editing). This class contains the functionality of the algorithm for synchronization described in Section 4.6.2.2.

The editing however, is implemented in a way making it content-type dependent. As seen in the UML diagram in Figure 5.4 there is one non-abstract editor class used for changing the structure of a SVG document called `SVGSXEController`. The class `SXEController` contains basic functionality for editing XML trees, while the SVG specific one contains the functionality specific for editing an SVG document. Adding library support for other content types would simply mean implement another editing *SXE controller*. An SXE controller also takes care of initiating of the basic document structure needed, for example the SVG specific controller initiates a document by adding elements describing an empty drawing space.

5.5 Porting to Android

The initial development was done using the standard Java edition from Sun Microsystems, and because the Java implementation used on the target platform Android differed slightly there was some compatibility issues needed to be taken care of. These issues was because certain parts of the Java 2 Standard Edition were missing. All the different libraries and toolkits used needed to be analyzed, and the outcome was that JGroups and Smack had to be altered. JGroups used `MXBean` and `MBean` from `javax.management` and `java.management`. However, the functionality these packages provided for JGroups was not used in the application described in this report, thus, removing the parts depending on these packages solved all compatibility problems with JGroups and Android.

One compatibility issues related to Smack were the usage of the package `java.beans`, however, the functions related to this part of the Smack library was not necessary for the application either, so they were removed. Another compatibility issue with Smack were the initiation procedure. Smack, by default, uses the `META-INF` directory of the *JAR* file to store an XML file containing information about what packages should be initiated. However, Android does not support providing static file content via this method, instead files are put in a specific directory and then accessed via Android specific functions. This was solved by implementing the ability to manually provide the Smack initialization part with a file stream object. This way while initiating the Android application, the programmer can provide an Android specific way to load the files Smack uses for initiation. This way of initiating would also make it work better on other embedded systems where similar restrictions can be found.

5.5.1 Android Issues

In the early steps while developing the android application there were some issues with the multicast implementation on the devices. In the beginning of the development, the version of Android used was 1.0. The development was done on three different platforms. One was using VirtualBox[4], another one was the official emulator and at last, using the HTC G1 Smartphone[5]. The issue that was brought up was a bug in the multicast code, causing the application to crash when calling the function `setMulticastInterface()`, which is usually called before joining a multicast group to specify what network interface to use. The workaround to avoid this issue was fairly easy; simply avoiding calling this function in JmDNS and JGroups (the two toolkits using multicast sockets) made the problem go away with acceptable consequences. However, after Android 1.5 was released, by upgrading the HTC G1 to the new version makes the multicast functionality stop completely. Since the development model used by Android is fairly open, an issue

[4]Virtualization application

[5]The first Android smartphone available as a consumer device.

report was filed, and during the time the development of the application was done, other developers reported they were affected by the same bug as well.

5.6 Android Application

Even though the Android application is the part of the application that would be visible for an end-user, this is the least important part of this report, technology wise. The fields that this report aims to be related to most is network programming and distributed systems, not user interfaces. However, to make use of the parts related to those two fields, there needs to be an interface, if it is a graphical application or any other way. The Android application is in this section referred as the user interface.

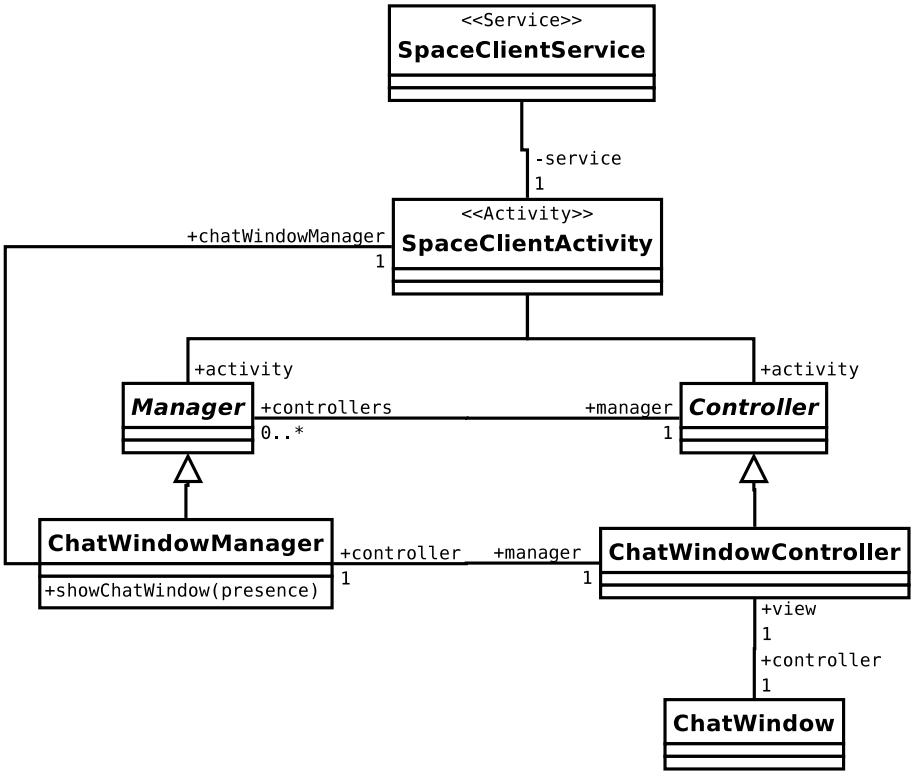
Some design choices that were made when implementing the user interface were that it should be able to run in the background as a multi purpose XMPP client. By using the client library developed for this application it was easy to design it in the way that adding support for server based communication would be easy.

As the interface is to be run on an Android phone, some considerations has to be made concerning this particular platform. For example, there are several things to deal with for minimizing the battery consumption. For example, as the application should have one part that runs in the background. An interface should not take any processing time when it is not visible (for example when the user temporarily starts using the web browser), but still should have the possibility to reestablish itself in the same state it was before going invisible. Thus, to address all these goals, the application was divided up into an Android *Service* and an Android *Activity*. A Service can be set up to run in the background until told to be turned of, while an Activity can be set up to only runs when it is visible. In the service part, the XMPP client is set up to run. It also contains user interface data such as active chats and chat histories, open collaboration spaces and similar things related to a user interface state.

The activity is the part of the interface started by the Android OS. When an activity is activated, it either connects to a service, or initiates one if there is no one to connect to. The activity contains all of the user interface related functionality, however, as mentioned, uses the service to save its state. The interface consists of some major parts: contact list - which lists the available presences, chat windows - sending and receiving XMPP messages to other presences and collaboration space windows. The implementation design is done in a way that every window is an Android *View*, and every window has its own *Controller*. If the windows can be considered as a part of a group, for example chat windows can be considered as a group of windows containing chat sessions, they have a *Manager* that keeps track of open windows.

Figure 5.5 shows a diagram of how managers, controllers and views are related. In the diagram there are three typical Android specific classes to be found; `SpaceClientService` - the background service running the XMPP client, `SpaceClientActivity` - containing

Figure 5.5: Android Application - Managers and Controllers



all the graphical functions related to the user interface, and `ChatWindow` - a specific part of the view.

6 Conclusion

After completing the work a working application running on Android had been implemented. Together with this application there a number of protocol specifications were written. The initial goal that a whiteboarding application communicating via an wireless ad-hoc network was downgraded to be using a more structured wireless network for example one using an wireless access point, however. Other than that, all the initial goals were fulfilled. By using only current day smartphones it was possible to perform collaborative whiteboarding without any major drawbacks, sharing a drawing space between a number of devices using a graphical interface.

It is shown that the hardware and software capabilities some of today's smartphones are enough for implementing third party applications with the functionality of the application this report discusses. However, it may not be concluded what limit the network method used has when it comes to number of users participating in a space.

It can also be considered that it was favorable to use and contribute to open source projects. Except the protocol specifications and the implementation specific to those as well as the Android application, everything related to the development regarding software in use, including work station, software development kit, target platform, toolkits as well as protocol specifications has been open source/free software or open standards. Added features to JmDNS and Smack were submitted to the respective communities and during the time the thesis project was done they were tested and used by various other developers. However, the other parts not related to existing open source toolkits was not published as open source and thus not tested or used by any third party users.

References

[1] Bengtsson, M. Memo: Synhcronization. June 2006.

[2] Birman, K., Constable, B., Hayden, M., Hickey, J., Kreitz, C., Renesse, R. V., Rodeh, O., and Vogels, W. The horus and ensemble projects: Accomplishments and limitations. *DARPA Information Survivability Conference and Exposition, 1* (2000), 0149.

[3] Birman, K. P. *Building secure and reliable network applications.* Manning Publications Co., Greenwich, CT, USA, 1997.

[4] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and Thayer, R. OpenPGP Message Format. RFC 4880 (Proposed Standard), Nov. 2007. Updated by RFC 5581.

[5] Cheshire, S., Aboba, B., and Guttman, E. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927 (Proposed Standard), May 2005.

[6] Cheshire, S., and Krochmal, M. DNS-Based Service Discovery. Internet-Draft draft-cheshire-dnsext-dns-sd-05, Internet Engineering Task Force, Sept. 2008. Work in progress.

[7] Cheshire, S., and Krochmal, M. Multicast DNS. Internet-Draft draft-cheshire-dnsext-multicastdns-07, Internet Engineering Task Force, Sept. 2008. Work in progress.

[8] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and Polk, W. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.

[9] Défago, X., Schiper, A., and Urbán, P. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Surveys (CSUR) 36*, 4 (2004), 372–421.

[10] Dierks, T., and Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008.

[11] Dollimore, J., Kindberg, T., and Coulouris, G. *Distributed Systems: Concepts and Design (Fourth Edition).* Addison Wesley, Boston, MA, USA, 2005.

[12] Eatmon, R., Hildebrand, J., Miller, J., Muldowney, T., and Saint-Andre, P. Data Forms. XMPP Extension XEP-0004, XMPP Standards Foundation, Aug. 2007. Final.

[13] Gidenstam, A., Koldehofe, B., Papatriantafilou, M., and Tsigas, P. Dynamic and fault-tolerant cluster management. In *Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 237–244.

[14] Gidenstam, A., Koldehofe, B., Papatriantafilou, M., and Tsigas, P. Lightweight causal cluster consistency. *Proceedings of the Conference on Innovative Internet Community Systems (I2CS 2005), Lecture Notes in Computer Science Vol. 3908, Springer Verlag, 2006. 3908, s. 17-28* (2005).

[15] Goland, Y. Y., Cai, T., Leach, P., and Gu, Y. Simple Service Discovery Protocol. Internet-Draft draft-cai-ssdp-v1-03, Internet Engineering Task Force, Oct. 1999. Outdated and expired.

[16] Govenius, J. Shared XML Editing. XMPP Extension, XMPP Standards Foundation, Jan. 2008. ProtoXEP.

[17] Gulbrandsen, A., Vixie, P., and Esibov, L. A DNS RR for specifying the location of services (DNS SRV). RFC 2782 (Proposed Standard), Feb. 2000.

[18] Guttman, E., Perkins, C., Veizades, J., and Day, M. Service Location Protocol, Version 2. RFC 2608 (Proposed Standard), June 1999. Updated by RFC 3224.

[19] Hayden, M., and Birman, K. Probabilistic broadcast. Tech. rep., Ithaca, NY, USA, Sept. 1996.

[20] Hildebrand, J., Millard, P., Eatmon, R., and Saint-Andre, P. Service Discovery. XMPP Extension XEP-0030, XMPP Standards Foundation, June 2008. Final.

[21] Hildebrand, J., and Saint-Andre, P. Field Standardization for Data Forms. XMPP Extension XEP-0068, XMPP Standards Foundation, July 2004. Active.

[22] Hildebrand, J., and Saint-Andre, P. Stream Compression. XMPP Extension XEP-0138, XMPP Standards Foundation, May 2009. Final.

[23] Hildebrand, J., Saint-Andre, P., Tronçon, R., and Konieczny, J. Entity Capabilities. XMPP Extension XEP-0115, XMPP Standards Foundation, Feb. 2008. Draft.

[24] Mavrogiannopoulos, N. Using OpenPGP Keys for Transport Layer Security (TLS) Authentication. RFC 5081 (Experimental), Nov. 2007.

[25] Meyer, D. C2C Authentication using TLS. XMPP Extension XEP-0250, XMPP Standards Foundation, Sept. 2008. Experimental.

[26] Mockapetris, P. Domain names - implementation and specification. RFC 1035 (Standard), Nov. 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343.

[27] Morill, D. Mail list post to XMPP standards concerning Android and XMPP compliance (http://mail.jabber.org/pipermail/standards/2008-February/018015.html), Nov. 2008.

[28] Preston, J. *Rethinking consistency management in real-time collaborative editing systems*. PhD thesis, Citeseer, 2007.

[29] Saint-Andre, P. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 3920 (Proposed Standard), Oct. 2004.

[30] Saint-Andre, P. End-to-End XML Streams. XMPP Extension XEP-0246, XMPP Standards Foundation, July 2008. Experimental.

[31] Saint-Andre, P. Serverless Messaging. XMPP Extension XEP-0174, XMPP Standards Foundation, Nov. 2008. Final.

[32] Taylor, D., Wu, T., Mavrogiannopoulos, N., and Perrin, T. Using the Secure Remote Password (SRP) Protocol for TLS Authentication. RFC 5054 (Informational), Nov. 2007.

Appendix

A Manual

The Manual for Shared Resource for Collaborative Editing over a Wireless Network

1 Compiling and Installing

Compiling

As the different parts of the application relies on each other in various ways, there is a preferred order which the different toolkits and libraries are to be compiled. The two toolkits that doesn't rely on any other non-packaged third party libraries are JGroups and JmDNS. Even though they are altered to fit the needs of the application of this report they can be compiled in the same way as described in the bundled installation instructions. They will both generate their own *JAR* files which then are used when compiling later packages.

Next step is the Smack XMPP library, which depends on JmDNS for its server-less components. This also is a set of added features to an existing toolkit and the existing compilation instructions are still usable. By following the instructions two *JAR* files will be generated; one containing the basic functionality of an XMPP client including the server-less component, the other one containing extended functionality such as service discovery and file transfer.

Both relying on the Smack library and the JGroups toolkit, the client library is compiled by using the *Ant* compilation toolkit. Having Ant installed, generating a complete *JAR* file is done by simply entering the project directory and running the command `ant jar`. Before doing this, the *JAR* files generated when compiling JmDNS and JGroups must be provided and placed in appropriate directories in the client library project tree.

The Android application is compiled also by using *Ant*, however, it has to be set up to be used with an Android SDK. It can be done by using Eclipse, but when developing this application, only Ant was used together with an Ant build file. Entering the project directory of the Android application, running the command `ant debug` would build, generate and sign a debug installation package, which could be used for installing in an emulator or on a physical device. As the Android application relies on the client library, the JAR file is, before compiling, needed to be provided and placed in the appropriate directory.

Installing

Installing the program is done by using the Android SDK and an USB cable or by providing a URL to an installation package. The installation process is done completely by the Android platform after the installation packet is provided.

2 Using

Running

To start the application the normal procedure for starting a program on an Android device is used, simply clicking on the icon. When activated, the user will be presented with a roster view, showing available presences.

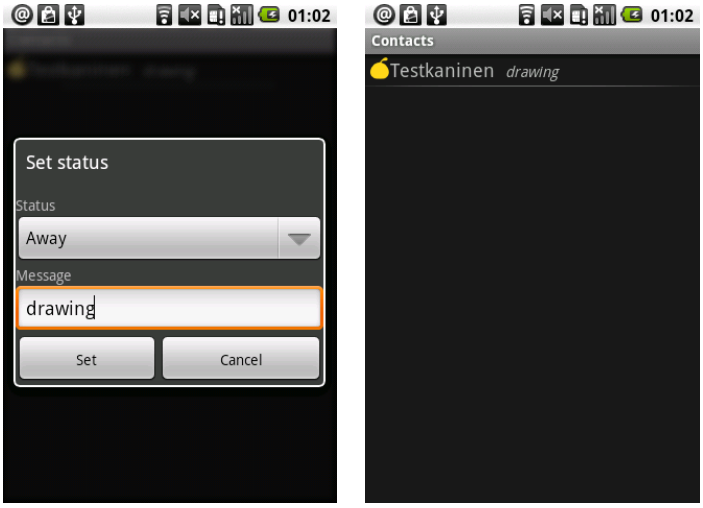
The roster

Figure 2.1: Screenshot - Roster view

Figure 2.1 shows how it may look. There is, at the version present at the time this report was written, no way to configure social information such as name, Jabber ID or E-Mail address. These data are all set to default values used for testing.

Status

Social statuses can be set. Figure 2.2a shows how to set the status of the user, and Figure 2.2b shows how a social status is displayed in the contact window. The status set using the dialog are also visible from other clients supporting serverless XMPP, and



(a) Editing (b) Viewing

Figure 2.2: Screenshot - Setting status

other clients specifying their statuses using serverless XMPP can be displayed in the contact window.

Chat Windows

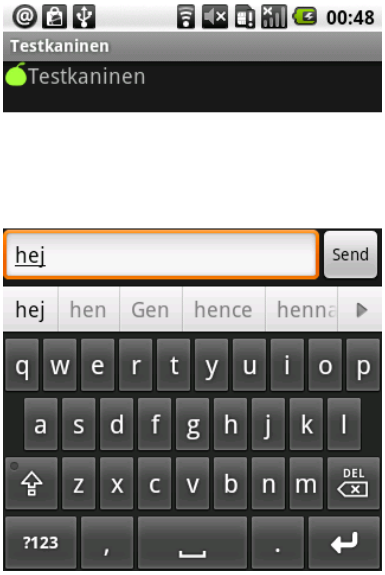
The chat windows are created one per presence, and accessed via a list of active chat windows, or via the roster view. Figure 2.3 shows a screenshot of a chat window with a presence, ready to send the first message.

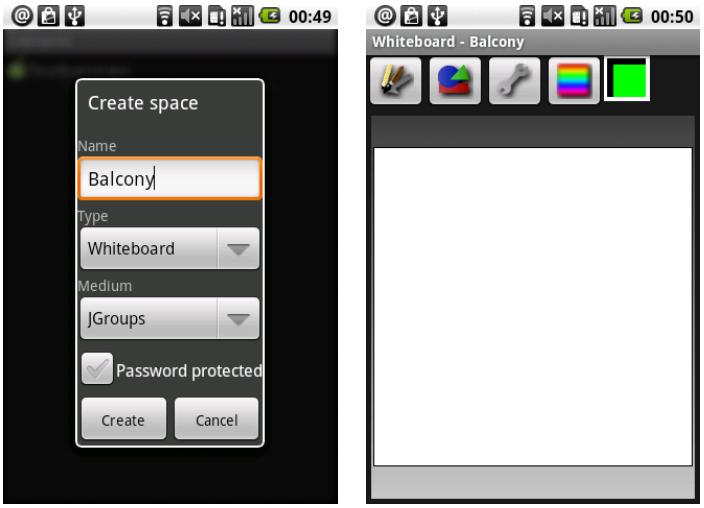
2.1 Collaboration Spaces

Creating

Collaboration spaces can be created and search for using the menu for spaces. By selecting the option for creating new spaces, the user is presented with a dialog for setting the preferences needed to create a new space. Figure 2.4 shows how this procedure works, and how a newly created whiteboard space looks like. The four icons above the drawing space shown in Figure 2.4b are used for accessing settings. The left most is used for changing tool, the second one for changing the shape to be drawn, the third one is for changing drawing mode or text element parameters. The fourth is used for changing the fill color or stroke color. To the right, next to the icons is an area showing the current color.

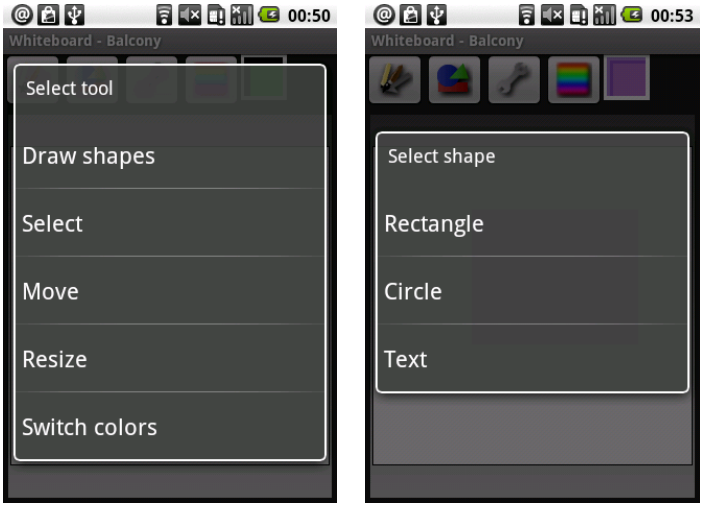
Figure 2.3: Screenshot - Chat window





(a) Preferences (b) Empty whiteboard

Figure 2.4: Screenshot - Creating a Whiteboard



(a) Tools (b) Shapes

Figure 2.5: Screenshot - Tools and Shapes

Tools and Shapes

Using the buttons on the top of the Whiteboard window one can access tools and shapes. Figure 2.5 shows available tools and shapes. The *Draw shapes* alternative is used for creating new objects, while the others are used to edit existing ones.

By selecting *Draw shapes* seen in Figure 2.5 the user can draw shapes by pressing on the screen, dragging specifying the size, and releasing finalizing the drawing operation. Figure 2.6 shows an example of when two objects are drawn, one rectangle and one circle. It also shows when colors have been changed, which is described in the next section.

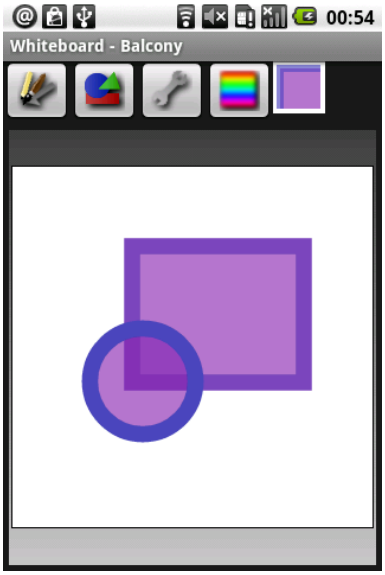
Entering and Editing Text

The user interface also has support for entering and editing text elements in the SVG document. Adding a text element is done by using the drawing tool then clicking on the drawing area. The user will be presented with a dialog (Figure 2.7a) and after pressing the *Create* button, the text element is created. Figure 2.7b illustrates how a work space may look after text has been edited. The drawing mode, colors and thickness of the border may be changed in the same way as a normal graphics element. It is also possible to change an existing text by first selecting using the *Select* tool seen in Figure 2.5 then going via the options menu selecting *Text Options*.

Colors and Modes

A user can set the way an object is to be drawn. One can specify if the border of a shape should be drawn, or if it should be filled with some kind of color, or possibly both. One can also set the width of the border. Figure 2.8 shows how drawing mode and colors can be changed.

Figure 2.6: Screenshot - Drawing



(a) Entering (b) Displaying

Figure 2.7: Screenshot - Text

There are also two possible colors to set, the *Stroke color* - color of the border of the shape, and *Fill color* - the color that the shape is filled with. When selecting a color the user can choose the Hue from a circle containing all posible hue values to the left. To the right there are also all available color parameters available: *Hue*, *Saturation*, *Value* and *Alpha*. *Alpha* is used to set transparency, when filled bar means not transparent.

(a) Drawing mode (b) Color

Figure 2.8: Screenshot - Drawing Modes

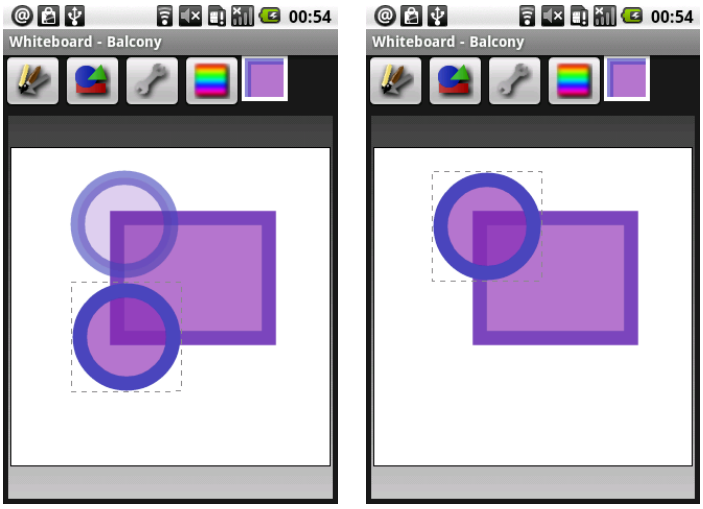
Moving Shapes

Figure 2.9 shows how to move an object. It is done by first selecting *Move* (seen in Figure 2.5) and then by pressing, dragging and releasing using the touch screen, objects can be moved. A half transparent preview is displayed while dragging, but when releasing (lifting up the finger) the object will be moved. The document structure is not altered until the object is released.

Resizing Shapes

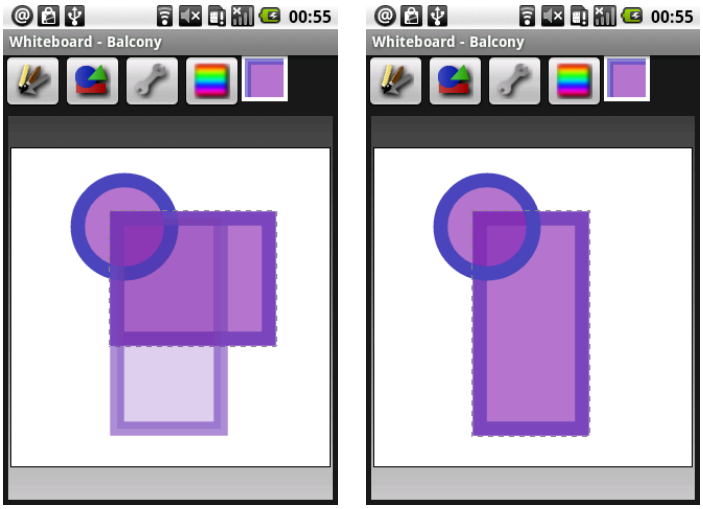
Resizing an object is shown in Figure 2.10. It is done similar to how moving is done, except it's only supported by a subset of the available shapes, rectangles and circles. Resizing a circle and rectangle is done using different techniques. Rectangles are resized by dragging (press, drag and release) one of its corner towards some direction. Only the corner that is closest to the position the press point will be changed. As moving, the real action is not done until releasing the object.

Circles are resized by dragging as well. The technique used for rectangles can naturally not be applied to circles, and instead of changing properties of cornerers, dragging simply changes the r property, the radius.



(a) Moving (b) Moved

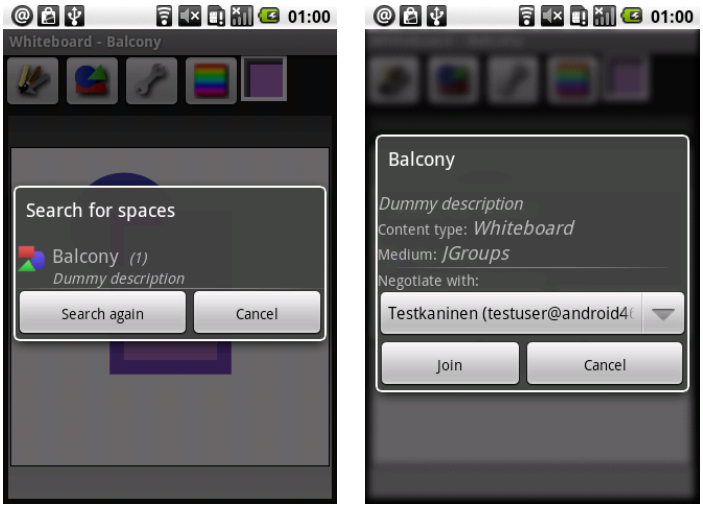
Figure 2.9: Screenshot - Moving Shapes



(a) Resizing (b) Resized

Figure 2.10: Screenshot - Resizing Shapes

Searching and Joining



(a) Searching (b) Joining

Figure 2.11: Screenshot - Searching and Joining

Searching and joining is performed by choosing the search option from the menu. When opening the dialog, a new search is automatically initiated, and new found space sessions are updated as they are discovered. Figure 2.11a shows how one space session has been found. When clicking on an element, another dialog containing a bit more detailed information is displayed to the user, as shown in Figure 2.11b. The user is also presented with the option to join the session by clicking the *Join* button. If a user chooses to join a new session, the application will connect and initialize the space, presenting the user with the new whiteboard (if the space is of the whiteboard type), and the user can then start edit the space together with the other participants. It is however not possible to join a space session twice, except if the session is canceled and the new join procedure is initiated after the cancellation is finished.

B XMPP Extension: Advertising Collaborative Work Spaces

XEP-XXXX: Collaborative Work Space Sessions

This specification defines how to discover and negotiate participation to a collaborative work spaces. It is ment to work without any sort of server interaction and is designed to be compatible with Serverless Messaging (XEP-0174)

Document Information

Series: XEP
Number: XXXX
Status: ProtoXEP
Type: Informational
Version: 0.3
Last Updated: 2009-07-20
Approving Body: XMPP Council
Dependencies: XMPP Core, XEP-0030, XEP-0174, RFC 3927, draft-cheshire-dnsext-dns-sd, draft-cheshire-dnsext-multicastdns
Supersedes: None
Superseded By: None
Short Name:

Relation to XMPP

The Extensible Messaging and Presence Protocol (XMPP) is defined in the XMPP Core (RFC 3920) and XMPP IM (RFC 3921) specifications contributed by the XMPP Standards Foundation to the Internet Standards Process, which is managed by the Internet Engineering Task Force in accordance with RFC 2026. Any protocol defined in this document has been developed outside the Internet Standards Process and is to be understood as an extension to XMPP rather than as an evolution, development, or modification of XMPP itself.

Conformance Terms

The following keywords as used in this document are to be interpreted as described in RFC 2119: "MUST", "SHALL", "REQUIRED"; "MUST NOT", "SHALL NOT"; "SHOULD", "RECOMMENDED"; "SHOULD NOT", "NOT RECOMMENDED"; "MAY", "OPTIONAL".

Table of Contents

---

1. Introduction

1.1 Motivation

Considering the occasion when a group of XMPP clients are all located in a global internet or a local area network (communicating with each other in a serverless manner (using XEP-0174)), there are times when there is a need for a common work space between some of the nodes, such as a common group chat, or a shared white board or in general any work space wich keeps synchronized between the clients. This extension describes how to discover and negotiate participation in a group communication where a document is collaboratively edited.

1.2 How It Works

Discovery and advertising of a ongoing group communication relies on Service Discovery (XEP-0030) for requesting advertised sessions, Entity Capabilities (XEP-0115) for advertising. Communication can be based on server based messaging or Serverless Messaging (XEP-0174) for peer-to-peer communication.

In the scenario where Serverless Messaging is used, the procedure for finding, requesting information and negotiating participation goes as follows. The two presences romeo@forza and juliet@pronto are aware of each other using Serverless Messaging. The entity romeo@forza is standing on the ground while juliet@pronto is standing on a balcony, and because of too much background noice (the non-radio wave kind) romeo@forza cannot communicate the way he wants. To make it easier, juliet@pronto sets up a collaborative work space and urges romeo@forza to find it. romeo@forza will then use the following procedure to discover and participate.

0. First romoe@forza discovers juliet@pronto's link-local presence. As described in XEP-0174, romeo@forza will know, by looking at the "node" and "ver" entries in the TXT field, or by using Service Discovry, if juliet@pronto supports this collaborative work space extension.
1. After drawing the conclusion that juliet@pronto supports this extension romeo@forza uses the XML stream initiated during service discovery or if the "node" and "ver" entries were already known; initiates a XML stream between them, as described in XEP-0174.
2. In this XML stream romeo@forza will, using the procedure described in this extension, request information about available sessions.
3. If romeo@forza finds a session suitable for what he is looking for, he may now, according to what is described in this extension, negotiate participation in the ongoing session.
4. In case that his participation is accepted, he may now communicate using the collaborative work space, such as drawing flowers and hearts on a common whiteboard.

A similar procedure can be used when in a server based network, where presences including "node" and "ver" data are discovered via <presence/> stanzas.

2. Discovery

2.1 Service Discovery

To determine if an entity supports this extension, the requester uses Service Discovery. The requester makes an "#info" query to the responder. If supported, the responder includes a <feature/> with the "var" of "urn:itri:xmpp:space".

Example 1. Disco request for information

```
<iq type='get'
    to='juliet@pronto'
    from='romeo@forza'
    id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Example 2. Disco result for information

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='disco'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <feature var='urn:itri:xmpp:space'/>
    ...
  </query>
</iq>
```

2.2 Entity Capabilities and Serverless Messaging considerations

If an entity wants to scan the local network for ongoing sessions, this process can be speeded up by using Entity Capabilities (XEP-0115).

2.3 Requesting Information

In case the expected feature is advertised romeo may continue requesting information about ongoing sessions. This is done by sending an <iq> request with the type "get", containing the element <spaces/> in the namespace "urn:itri:xmpp:space".

Example 3. Requesting information about current sessions

```
<iq type='get'
    to='juliet@pronto'
    from='romeo@forza'
    id='discover1'>
  <spaces xmlns='urn:itri:xmpp:space'/>
</iq>
```

Example 4. Responding information about current sessions

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='discover1'>
  <spaces xmlns='urn:itri:xmpp:space'>
    <ad uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'>
      <description>
        <name>Balcony Illustrations</name>
        <info>Clarification of intentions</info>
        <medium>jgroups</medium>
        <conten-type>image/xml+svg</content-type>
        <members>1</members>
      </description>
    </ad>
    <ad uuid='b00c4ee3-0232-48c1-977a-fa3982beab0f'>
      <description>
        <name>Capulet Family Business</name>
        <medium>jgroups</medium>
        <content-type>text/plain</content-type>
        <negotiator>father@capulet</negotiator>
        <members>4</members>
        <authentication>form</authentication>
      </description>
    </ad>
  </spaces>
</iq>
```

3. Negotiating Participation

After discovering a session a user my try to negotiate participation with the user responsible, the advertiser or the one specified in the <negotiator/> element.

3.1 No authentication

When trying to join a session which donnot require any authentication, for example a casually set up white board for explaining something using simple graphical figures, there is no need for encryption, passphrase or similar. This can be done as romeo@forza does in the following example. He has already discovered the presence of juliet@pronto and found the session named "Balcony Illustrations" that he has decided to join. He first asks to join.

Example 5. Joining a casual session

```
<iq type='set'
    to='juliet@pronto'
    from='romeo@forza'
    id='join1'>
  <space xmlns='urn:itri:xmpp:space'
         state='join'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'/>
</iq>
```

After juliet@pronto received the request, her client MAY ask if she accepts the request or not. If she does accept romeo@forza's participation in the session accociated with the specified uuid, in this case "Balcony Illustrations", she will reply with the following stanza.

NOTICE: The medium element used in the following example are specified in another extension called "XMPP-Based Decentralized Group Communication using JGroups"

Example 6. Accepted to joining a casual session

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='join1'>
  <space xmlns='urn:itri:xmpp:space'
         state='result'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'>
```

```
    <medium xmlns='urn:itri:xmpp:space:medium:jgroups'>
      <cluster>juliet@pronto:f13f9f26-cce9-4171-80d8-e251c67bb0a1</cluster>
    </medium>
  </space>
</iq>
```

If, in fact, juliet@pronto does NOT want romeo@forza to participate in the she will reply with the following stanza.

Example 7. Denied to joining a casual session

```
<iq type='error'
    from='juliet@pronto'
    to='romeo@forza'
    id='join1'>
  <error code='401' type='auth'>
    <not-authorized xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

3.2 Joining a session requiring filling out a form

Considering the case that juliet@pronto wants to put some restrictions on the work space she created she can add pass phrase protection by requiring the joining party to fill out a form containing for example a pass phrase. For romeo@forza to become a participator he has to have knowledge about this shared secret before joining.

Example 8. Joining a session requiring filling out a form #1

```
<iq type='set'
    to='juliet@pronto'
    from='romeo@forza'
    id='join1'>
  <space xmlns='urn:itri:xmpp:space'
         state='join'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'/>
</iq>
```

Now, juliet@pronto may respond with a form that romeo@forza is to fill in.

Example 9. Joining a session requiring filling out a form #2

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='join1'>
  <space xmlns='urn:itri:xmpp:space'
         state='challenge'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'>
    <challenge xmlns='urn:itri:xmpp:space:challenge:form'>
      <x xmlns='jabber:x:data'
         type='form'>
        <title>Capulet's orchard.</title>
        <instructions>My ears have not yet drunk a hundred words
Of that tongue's utterance, yet I know the sound:
Art thou not Romeo and a Montague?</instructions>
        <field type='text'
               label='Pass phrase'
               var='passphrase'/>
      </x>
    </challenge>
  </space>
</iq>
```

romeo@forza now has to fill in the form in order for juliet@pronto to accept his participation. He does this by sending the following stanza.

Example 10. Joining a session requiring filling out a form #3

```
<iq type='set'
    to='juliet@pronto'
    from='romeo@forza'
    id='join2'>
  <space xmlns='urn:itri:xmpp:space'
         state='join'
```

```
          role='participator'
          uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'>
      <response xmlns='urn:itri:xmpp:space:challenge:form'>
        <x xmlns='jabber:x:data'
           type='submit'>
          <field type='text' var='passphrase'>
            <value>Neither, fair saint, if either thee dislike.</value>
          </field>
        </x>
      </response>
    </space>
  </iq>
```

As in the case of joining a casual session juliet@pronto may deny or accept the joining, depending on the input given by romeo@forza. In case of accepting, she will reply the following.

NOTICE: The medium element used in the following example are specified in another extension called "XMPP-Based Decentralized Group Communication using JGroups"

Example 11. Joining a session requiring filling out a form #4

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='join2'>
  <space xmlns='urn:itri:xmpp:space'
         state='result'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'/>
    <medium xmlns='urn:itri:xmpp:space:medium:jgroups'>
      <cluster>juliet@pronto:f13f9f26-cce9-4171-80d8-e251c67bb0a1</cluster>
    </medium>
  </space>
</iq>
```

If juliet@pronto still doesn't want romeo@forza to participate in the session, she may reply with an error.

Example 12. Joining a session requiring filling out a form #5

```
<iq type='error'
    from='juliet@pronto'
    to='romeo@forza'
    id='join2'>
  <error code='401' type='auth'>
    <not-authorized xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

4. Collaboration

Collaboration between entities is performed by Collaborative XML Editing.

5. Formal Description

New elements introduced in this extension are <spaces/> and <space/>. The following section describes how these are used.

5.1 <spaces/> Element

The <spaces> contains advertisements of ongoing sessions a client is participating in. The namespace "urn:itri:xmpp:space" MUST be specified.

5.2 <ad/> Element

The <ad/> provides information about an ongoing session. The attribute "uuid" MUST be included. The "uuid" attribute specifies the Universal Unique IDentifier for the ongoing session. An <ad/> element SHOULD provide descriptive information about the session in a sub element <description/>.

5.3 <description/> Element

The sub elements of the <description/> element contains description and properties of an ongoing session. The following table describes what the available elements and their values are.

Table 1: Sub elements to the <description/> Element

Element	Value	Description

Element	type	Description
authentication	string	Authentication method. Allowed values described beneath.
medium	string	The method used for performing Group Communication, possibly outside of the XMPP scope. Possible values are described later.
members	integer	The number of participants the moment the request was received. MUST be 1 or more.
name	string	The name or title of a space. May be, depending on the content, type be a part of the document.
info	string	An informative string describing what the space contains. May be, depending on the content type, be a part of the document.
negotiator	string	The Full JID or Service Name of the user responsible for negotiating participation. If non is specified the user advertising can act as a negotiator.
content-type	string	The type of the content, for exmaple image/svg+xml.
public	boolean	If true, anyone can view the work space, but not necessarily edit.

The authentication element specifies what kind of authentication is required. If it's not included, the implementation SHOULD read this as no authentication is required. Possible authentication mechanisms follows.

Table 2: Authentication mechanisms

Mechanism	Description
form	The joining user will receive a challenge in form of a form to fill in. The form is represented as a Data Form (XEP-0004) and can request any relevant information. It may for example be used if there is a shared secret required to join.

5.4 <space/> Element

The <space/> element is used during negotiation participation in an ongoing session. The attributes "uuid" and "state" MUST be included. The "uuid" attribute specifies the Universal Unique IDentifier for the ongoing session. The "state" attribute describes what state it is in (available states follows). The namespace MUST be set to "urn:itri:xmpp:space". A table of possible attributes and descriptions follows.

The attribute "state" specifies what state of element this <space/> is. The following table describes all possible values.

Table 3: Available "state" values

State	Description
join	An attempt to join a session. May contain no sub elements. If more are required for participating the other party will respond with a challenge. If the request contains additional information required to join, this information MUST be represented as a child element named <response/> and a namespace MUST be specified with it.
challenge	Sent by the party the joining client is negotiating with, if more data is required to join. A <space/> element with this state MUST include a <challenge/> element as a child element. The <challenge/> element MUST specify a namespace corresponding to what kind of challenge it is.
result	The result state contains information needed to start participating in the session. A <space/> element with this state MUST contain a <medium/> element as a child element. The <medium/> element MUST specify a namespace corresponding to what kind of medium it is.

5.5 <challenge/> Element

This element contains a challenge by the entity the joining party are negotiating with in order to verify that a join is to be allowed. The following table lists available namespaces for different authentication methods.

Table 4: Challenge types

Namespace	Short name	Description
urn:itri:xmpp:space:challenge:form	form	The challenge method MUST contain a <x/> element of type "form" specified in XEP-0004 (Data Forms). This joining party is required to fill in this form, which may contain fields

		such as a secret pass phrase, username or other required data. A set of predefined form types are described later.

5.6 <response/> Element

This element contains the response to the challenge sent by the host party (the party taking care of authorizing participation). The following table lists available namespaces for different authentication methods.

Table 5: Response types

Namespace	Short name	Description
urn:itri:xmpp:space:challenge:form	form	The response MUST contain a <x/> element of type "submit", specified in XEP-0004 (Data Forms) and MUST be a filled out form receieved in the challenge.

5.7 <medium/> Element

In the "result" state <space/> element one will find this element. It describes how to access the collaborative work space using the information described within. Different namespace reperesent different kind of mediums, and the following table describes available ones.

Table 6: Medium types

Namespace	Short name	Description
urn:itri:xmpp:space:medium:jgroups	jgroups	Described in the extension XMPP-Based Decentralized Group Communication using JGroups

5.8 Predefined Challenge form types

Here follows a set of predefined form types (FORM_TYPE) that may be used. Any form can be used, but a set of predefined types can make it easier for user interface developers to create better looking interfaces.

5.8.1 Password form

```
<x xmlns='jabber:x:data' type='form'>
  <instructions>Enter the correct password to access the collaborative work space.</instructions>
  <field var='FORM_TYPE' type='hidden'>
    <value>urn:itri:xmpp:space:challenge:form:pw</value>
  </field>
  <field label='Enter password'
         var='password'
         type='text-private'>
    <required/>
  </field>
</x>
```

6. Glossary

Table 7: Terminology

Term	Description
DNS-SD	A convention for naming and structuring DNS SRV records such that a client can dynamically discover a domain for a service using only standard DNS queries. See **draft-cheshire-dnsext-dns-sd**. For a full list of registered DNS-SD records, see <http://www.dns-sd.org/ServiceTypes.html>.
Multicast DNS (mDNS)	A technology that provides the ability to perform DNS-like operations on a local link in the absence of any conventional unicast DNS server. See **draft-cheshire-dnsext-multicastdns**.
JGroups	An Open Source project implementing a general suit for performing Group Communication. It's written in Java and licensed under LGPL. See <http://www.jgroups.org/>.

7. Acknowledgements

Notes

Revision History

Version 0.3 (2009-07-20)

Title more general and JGroups part is moved to it's own extension.

(jå)

Version 0.2 (2009-05-04)

Changed mode to content-type and added info-tag to description.

(jå)

Version 0.1 (2008-12-03)

The initial version.

(jå)

END

C XMPP Extension: Collaborative Editing using JGroups

XEP-XXXX: Collaborative Work Space Sessions

This specification defines how to discover and negotiate participation to a collaborative work spaces. It is ment to work without any sort of server interaction and is designed to be compatible with Serverless Messaging (XEP-0174)

Document Information

Series: XEP
Number: XXXX
Status: ProtoXEP
Type: Informational
Version: 0.3
Last Updated: 2009-07-20
Approving Body: XMPP Council
Dependencies: XMPP Core, XEP-0030, XEP-0174, RFC 3927, draft-cheshire-dnsext-dns-sd, draft-cheshire-dnsext-multicastdns
Supersedes: None
Superseded By: None
Short Name:

Relation to XMPP

The Extensible Messaging and Presence Protocol (XMPP) is defined in the XMPP Core (RFC 3920) and XMPP IM (RFC 3921) specifications contributed by the XMPP Standards Foundation to the Internet Standards Process, which is managed by the Internet Engineering Task Force in accordance with RFC 2026. Any protocol defined in this document has been developed outside the Internet Standards Process and is to be understood as an extension to XMPP rather than as an evolution, development, or modification of XMPP itself.

Conformance Terms

The following keywords as used in this document are to be interpreted as described in RFC 2119: "MUST", "SHALL", "REQUIRED"; "MUST NOT", "SHALL NOT"; "SHOULD", "RECOMMENDED"; "SHOULD NOT", "NOT RECOMMENDED"; "MAY", "OPTIONAL".

Table of Contents

---

1. Introduction

1.1 Motivation

Considering the occasion when a group of XMPP clients are all located in a global internet or a local area network (communicating with each other in a serverless manner (using XEP-0174)), there are times when there is a need for a common work space between some of the nodes, such as a common group chat, or a shared white board or in general any work space wich keeps synchronized between the clients. This extension describes how to discover and negotiate participation in a group communication where a document is collaboratively edited.

1.2 How It Works

Discovery and advertising of a ongoing group communication relies on Service Discovery (XEP-0030) for requesting advertised sessions, Entity Capabilities (XEP-0115) for advertising. Communication can be based on server based messaging or Serverless Messaging (XEP-0174) for peer-to-peer communication.

In the scenario where Serverless Messaging is used, the procedure for finding, requesting information and negotiating participation goes as follows. The two presences romeo@forza and juliet@pronto are aware of each other using Serverless Messaging. The entity romeo@forza is standing on the ground while juliet@pronto is standing on a balcony, and because of too much background noice (the non-radio wave kind) romeo@forza cannot communicate the way he wants. To make it easier, juliet@pronto sets up a collaborative work space and urges romeo@forza to find it. romeo@forza will then use the following procedure to discover and participate.

0. First romoe@forza discovers juliet@pronto's link-local presence. As described in XEP-0174, romeo@forza will know, by looking at the "node" and "ver" entries in the TXT field, or by using Service Discovry, if juliet@pronto supports this collaborative work space extension.
1. After drawing the conclusion that juliet@pronto supports this extension romeo@forza uses the XML stream initiated during service discovery or if the "node" and "ver" entries were already known; initiates a XML stream between them, as described in XEP-0174.
2. In this XML stream romeo@forza will, using the procedure described in this extension, request information about available sessions.
3. If romeo@forza finds a session suitable for what he is looking for, he may now, according to what is described in this extension, negotiate participation in the ongoing session.
4. In case that his participation is accepted, he may now communicate using the collaborative work space, such as drawing flowers and hearts on a common whiteboard.

A similar procedure can be used when in a server based network, where presences including "node" and "ver" data are discovered via <presence/> stanzas.

2. Discovery

2.1 Service Discovery

To determine if an entity supports this extension, the requester uses Service Discovery. The requester makes an "#info" query to the responder. If supported, the responder includes a <feature/> with the "var" of "urn:itri:xmpp:space".

Example 1. Disco request for information

```
<iq type='get'
    to='juliet@pronto'
    from='romeo@forza'
    id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Example 2. Disco result for information

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='disco'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <feature var='urn:itri:xmpp:space'/>
    ...
  </query>
</iq>
```

2.2 Entity Capabilities and Serverless Messaging considerations

If an entity wants to scan the local network for ongoing sessions, this process can be speeded up by using Entity Capabilities (XEP-0115).

2.3 Requesting Information

In case the expected feature is advertised romeo may continue requesting information about ongoing sessions. This is done by sending an <iq> request with the type "get", containing the element <spaces/> in the namespace "urn:itri:xmpp:space".

Example 3. Requesting information about current sessions

```
<iq type='get'
    to='juliet@pronto'
    from='romeo@forza'
    id='discover1'>
  <spaces xmlns='urn:itri:xmpp:space'/>
</iq>
```

Example 4. Responding information about current sessions

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='discover1'>
  <spaces xmlns='urn:itri:xmpp:space'>
    <ad uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'>
      <description>
        <name>Balcony Illustrations</name>
        <info>Clarification of intentions</info>
        <medium>jgroups</medium>
        <conten-type>image/xml+svg</content-type>
        <members>1</members>
      </description>
    </ad>
    <ad uuid='b00c4ee3-0232-48c1-977a-fa3982beab0f'>
      <description>
        <name>Capulet Family Business</name>
        <medium>jgroups</medium>
        <content-type>text/plain</content-type>
        <negotiator>father@capulet</negotiator>
        <members>4</members>
        <authentication>form</authentication>
      </description>
    </ad>
  </spaces>
</iq>
```

3. Negotiating Participation

After discovering a session a user my try to negotiate participation with the user responsible, the advertiser or the one specified in the <negotiator/> element.

3.1 No authentication

When trying to join a session which donnot require any authentication, for example a casually set up white board for explaining something using simple graphical figures, there is no need for encryption, passphrase or similar. This can be done as romeo@forza does in the following example. He has already discovered the presence of juliet@pronto and found the session named "Balcony Illustrations" that he has decided to join. He first asks to join.

Example 5. Joining a casual session

```
<iq type='set'
    to='juliet@pronto'
    from='romeo@forza'
    id='join1'>
  <space xmlns='urn:itri:xmpp:space'
         state='join'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'/>
</iq>
```

After juliet@pronto received the request, her client MAY ask if she accepts the request or not. If she does accept romeo@forza's participation in the session accociated with the specified uuid, in this case "Balcony Illustrations", she will reply with the following stanza.

NOTICE: The medium element used in the following example are specified in another extension called "XMPP-Based Decentralized Group Communication using JGroups"

Example 6. Accepted to joining a casual session

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='join1'>
  <space xmlns='urn:itri:xmpp:space'
         state='result'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'>
```

```
    <medium xmlns='urn:itri:xmpp:space:medium:jgroups'>
      <cluster>juliet@pronto:f13f9f26-cce9-4171-80d8-e251c67bb0a1</cluster>
    </medium>
  </space>
</iq>
```

If, in fact, juliet@pronto does NOT want romeo@forza to participate in the she will reply with the following stanza.

Example 7. Denied to joining a casual session

```
<iq type='error'
    from='juliet@pronto'
    to='romeo@forza'
    id='join1'>
  <error code='401' type='auth'>
    <not-authorized xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

3.2 Joining a session requiring filling out a form

Considering the case that juliet@pronto wants to put some restrictions on the work space she created she can add pass phrase protection by requiring the joining party to fill out a form containing for example a pass phrase. For romeo@forza to become a participator he has to have knowledge about this shared secret before joining.

Example 8. Joining a session requiring filling out a form #1

```
<iq type='set'
    to='juliet@pronto'
    from='romeo@forza'
    id='join1'>
  <space xmlns='urn:itri:xmpp:space'
         state='join'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'/>
</iq>
```

Now, juliet@pronto may respond with a form that romeo@forza is to fill in.

Example 9. Joining a session requiring filling out a form #2

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='join1'>
  <space xmlns='urn:itri:xmpp:space'
         state='challenge'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'>
    <challenge xmlns='urn:itri:xmpp:space:challenge:form'>
      <x xmlns='jabber:x:data'
         type='form'>
        <title>Capulet's orchard.</title>
        <instructions>My ears have not yet drunk a hundred words
Of that tongue's utterance, yet I know the sound:
Art thou not Romeo and a Montague?</instructions>
        <field type='text'
               label='Pass phrase'
               var='passphrase'/>
      </x>
    </challenge>
  </space>
</iq>
```

romeo@forza now has to fill in the form in order for juliet@pronto to accept his participation. He does this by sending the following stanza.

Example 10. Joining a session requiring filling out a form #3

```
<iq type='set'
    to='juliet@pronto'
    from='romeo@forza'
    id='join2'>
  <space xmlns='urn:itri:xmpp:space'
         state='join'
```

```
          role='participator'
          uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'>
      <response xmlns='urn:itri:xmpp:space:challenge:form'>
        <x xmlns='jabber:x:data'
           type='submit'>
          <field type='text' var='passphrase'>
            <value>Neither, fair saint, if either thee dislike.</value>
          </field>
        </x>
      </response>
    </space>
  </iq>
```

As in the case of joining a casual session juliet@pronto may deny or accept the joining, depending on the input given by romeo@forza. In case of accepting, she will reply the following.

NOTICE: The medium element used in the following example are specified in another extension called "XMPP-Based Decentralized Group Communication using JGroups"

Example 11. Joining a session requiring filling out a form #4

```
<iq type='result'
    from='juliet@pronto'
    to='romeo@forza'
    id='join2'>
  <space xmlns='urn:itri:xmpp:space'
         state='result'
         role='participator'
         uuid='f13f9f26-cce9-4171-80d8-e251c67bb0a1'/>
    <medium xmlns='urn:itri:xmpp:space:medium:jgroups'>
      <cluster>juliet@pronto:f13f9f26-cce9-4171-80d8-e251c67bb0a1</cluster>
    </medium>
  </space>
</iq>
```

If juliet@pronto still doesn't want romeo@forza to participate in the session, she may reply with an error.

Example 12. Joining a session requiring filling out a form #5

```
<iq type='error'
    from='juliet@pronto'
    to='romeo@forza'
    id='join2'>
  <error code='401' type='auth'>
    <not-authorized xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

4. Collaboration

Collaboration between entities is performed by Collaborative XML Editing.

5. Formal Description

New elements introduced in this extension are <spaces/> and <space/>. The following section describes how these are used.

5.1 <spaces/> Element

The <spaces> contains advertisements of ongoing sessions a client is participating in. The namespace "urn:itri:xmpp:space" MUST be specified.

5.2 <ad/> Element

The <ad/> provides information about an ongoing session. The attribute "uuid" MUST be included. The "uuid" attribute specifies the Universal Unique IDentifier for the ongoing session. An <ad/> element SHOULD provide descriptive information about the session in a sub element <description/>.

5.3 <description/> Element

The sub elements of the <description/> element contains description and properties of an ongoing session. The following table describes what the available elements and their values are.

Table 1: Sub elements to the <description/> Element

Element	Value	Description

Element	type	Description
authentication	string	Authentication method. Allowed values described beneath.
medium	string	The method used for performing Group Communication, possibly outside of the XMPP scope. Possible values are described later.
members	integer	The number of participants the moment the request was received. MUST be 1 or more.
name	string	The name or title of a space. May be, depending on the content, type be a part of the document.
info	string	An informative string describing what the space contains. May be, depending on the content type, be a part of the document.
negotiator	string	The Full JID or Service Name of the user responsible for negotiating participation. If non is specified the user advertising can act as a negotiator.
content-type	string	The type of the content, for exmaple image/svg+xml.
public	boolean	If true, anyone can view the work space, but not necessarily edit.

The authentication element specifies what kind of authentication is required. If it's not included, the implementation SHOULD read this as no authentication is required. Possible authentication mechanisms follows.

Table 2: Authentication mechanisms

Mechanism	Description
form	The joining user will receive a challenge in form of a form to fill in. The form is represented as a Data Form (XEP-0004) and can request any relevant information. It may for example be used if there is a shared secret required to join.

5.4 <space/> Element

The <space/> element is used during negotiation participation in an ongoing session. The attributes "uuid" and "state" MUST be included. The "uuid" attribute specifies the Universal Unique IDentifier for the ongoing session. The "state" attribute describes what state it is in (available states follows). The namespace MUST be set to "urn:itri:xmpp:space". A table of possible attributes and descriptions follows.

The attribute "state" specifies what state of element this <space/> is. The following table describes all possible values.

Table 3: Available "state" values

State	Description
join	An attempt to join a session. May contain no sub elements. If more are required for participating the other party will respond with a challenge. If the request contains additional information required to join, this information MUST be represented as a child element named <response/> and a namespace MUST be specified with it.
challenge	Sent by the party the joining client is negotiating with, if more data is required to join. A <space/> element with this state MUST include a <challenge/> element as a child element. The <challenge/> element MUST specify a namespace corresponding to what kind of challenge it is.
result	The result state contains information needed to start participating in the session. A <space/> element with this state MUST contain a <medium/> element as a child element. The <medium/> element MUST specify a namespace corresponding to what kind of medium it is.

5.5 <challenge/> Element

This element contains a challenge by the entity the joining party are negotiating with in order to verify that a join is to be allowed. The following table lists available namespaces for different authentication methods.

Table 4: Challenge types

Namespace	Short name	Description
urn:itri:xmpp:space:challenge:form	form	The challenge method MUST contain a <x/> element of type "form" specified in XEP-0004 (Data Forms). This joining party is required to fill in this form, which may contain fields

		such as a secret pass phrase, username or other required data. A set of predefined form types are described later.

5.6 <response/> Element

This element contains the response to the challenge sent by the host party (the party taking care of authorizing participation). The following table lists available namespaces for different authentication methods.

Table 5: Response types

Namespace	Short name	Description
urn:itri:xmpp:space:challenge:form	form	The response MUST contain a <x/> element of type "submit", specified in XEP-0004 (Data Forms) and MUST be a filled out form receieved in the challenge.

5.7 <medium/> Element

In the "result" state <space/> element one will find this element. It describes how to access the collaborative work space using the information described within. Different namespace reperesent different kind of mediums, and the following table describes available ones.

Table 6: Medium types

Namespace	Short name	Description
urn:itri:xmpp:space:medium:jgroups	jgroups	Described in the extension XMPP-Based Decentralized Group Communication using JGroups

5.8 Predefined Challenge form types

Here follows a set of predefined form types (FORM_TYPE) that may be used. Any form can be used, but a set of predefined types can make it easier for user interface developers to create better looking interfaces.

5.8.1 Password form

```
<x xmlns='jabber:x:data' type='form'>
  <instructions>Enter the correct password to access the collaborative work space.</instructions>
  <field var='FORM_TYPE' type='hidden'>
    <value>urn:itri:xmpp:space:challenge:form:pw</value>
  </field>
  <field label='Enter password'
         var='password'
         type='text-private'>
    <required/>
  </field>
</x>
```

6. Glossary

Table 7: Terminology

Term	Description
DNS-SD	A convention for naming and structuring DNS SRV records such that a client can dynamically discover a domain for a service using only standard DNS queries. See **draft-cheshire-dnsext-dns-sd**. For a full list of registered DNS-SD records, see <http://www.dns-sd.org/ServiceTypes.html>.
Multicast DNS (mDNS)	A technology that provides the ability to perform DNS-like operations on a local link in the absence of any conventional unicast DNS server. See **draft-cheshire-dnsext-multicastdns**.
JGroups	An Open Source project implementing a general suit for performing Group Communication. It's written in Java and licensed under LGPL. See <http://www.jgroups.org/>.

7. Acknowledgements

Notes

Revision History

Version 0.3 (2009-07-20)

Title more general and JGroups part is moved to it's own extension.

(jå)

Version 0.2 (2009-05-04)

Changed mode to content-type and added info-tag to description.

(jå)

Version 0.1 (2008-12-03)

The initial version.

(jå)

END

D XMPP Extension: Collaborative Editing of SVG Documents

XEP-XXXX: Collaborative XML Editing of SVG documents

This document specifies how Scalable Vector Graphics (SVG) can be used together with Collaborative XML Editing and Collaborative Work Space Sessions.

Document Information

Series: XEP
Number: XXXX
Status: ProtoXEP
Type: Informational
Version: 0.1
Last Updated: 2009-07-21
Approving Body: XMPP Council
Dependencies: XMPP Core, XEP-0030, XEP-0174
Supersedes: None
Superseded By: None
Short Name:

Relation to XMPP

The Extensible Messaging and Presence Protocol (XMPP) is defined in the XMPP Core (RFC 3920) and XMPP IM (RFC 3921) specifications contributed by the XMPP Standards Foundation to the Internet Standards Process, which is managed by the Internet Engineering Task Force in accordance with RFC 2026. Any protocol defined in this document has been developed outside the Internet Standards Process and is to be understood as an extension to XMPP rather than as an evolution, development, or modification of XMPP itself.

Conformance Terms

The following keywords as used in this document are to be interpreted as described in RFC 2119: "MUST", "SHALL", "REQUIRED"; "MUST NOT", "SHALL NOT"; "SHOULD", "RECOMMENDED"; "SHOULD NOT", "NOT RECOMMENDED"; "MAY", "OPTIONAL".

Table of Contents

---

1. Introduction

1.1 Motivation

The basic method for communication using computers are text messages. Some times communicating only using text may be too limited and some times drawing may be what is needed to more completely share thoughts. This protocol extension specifies how Scalable Vector Graphics (SVG) is used in combination with the Collaborative XML Editing extension for editing a shared document and Collaborative Work Space Sessions for discovery, advertisement and participation.

1.2 How it works

This document specifies how a Collaborative XML Editing session is initialized for using with a SVG 1.2 Tiny document, as well as mapping certain XML elements of SVG 1.2 Tiny to the description advertised in Collaborative Work Space Sessions.

2. Initiating

A client initiating a new collaborative XML editing session MUST initate the state to contain a number of required elements and attributes.

Elements that MUST be initiated is an the ones of a basic SVG document. It SHOULD contain the "title" element and the "desc" element. An example follows where the view box is set to be 800 pixels wide, 600 pixels high and start on the coordinates (0, 0).

Example 1. Basic SVG 1.2 Tiny document

```
<svg xmlns="http://www.w3.org/2000/svg"
     version="1.2" baseProfile="tiny"
     viewBox="0 0 800 600">
  <title>Balcony Illustrations</title>
  <desc>Clarification of intentions</desc>
</svg>
```

The content type of the document MUST be set to "image/xml+svg".

3. Advertisement

Advertisement MAY be done by using the Collaborative Work Space Sessions extension. If this is done the content of the element "title" SHOULD be advertised in the "name" element in the <description/> element, and the content of "desc" SHOULD be advertised in the "info" element in the <description/> element. The example given in the previous section could result in the following advertisement element.

Example 2. Advertisement of an SVG document

```
<description>
  <name>Balcony Illustrations</name>
  <info>Clarification of intentions</info>
  <conten-type>image/xml+svg</content-type>
</description>
```

Notes

Revision History

Version 0.1 (2009-07-21)

The initial version.

(jå)

END

E Code Documentation: Smack (Link-Local related parts)

Smack (Link-local related part)

Generated by Doxygen 1.6.1

Sun Dec 6 15:28:28 2009

Contents

1 Namespace Documentation

1.1 Package org.jivesoftware.smack

Classes

- class AbstractConnection
- class AbstractPacketReader

  Listens for XML traffic from the XMPP server and parses it into packet objects.

- class JmDNSPresenceDiscoverer

  An implementation of LLPresenceDiscoverer using JmDNS.

- class JmDNSService

  Implements a LLService using JmDNS.

- class LLChat

  Keeps track of a chat session between two link-local clients.

- interface LLChatListener

  Notification about new Link-local chat sessions.

- class LLConnectionConfiguration

  Link-local connection configuration settings.

- interface LLConnectionListener

  Notification about when new Link-local connections has been established.

- interface LLMessageListener

  Notification when messages are being delivered to a chat.

- class LLPacketReader

  Listens for XML traffic from a remote XMPP client and parses it into packet objects.

- class LLPresence

  Class for describing a Link-local presence information according to XEP-0174.

- class LLPresenceDiscoverer

  Link-local presence discoverer.

- interface LLPresenceListener

  Interface for receiving notifications about presence changes.

- class LLService

LLService acts as an abstract interface to a Link-local XMPP service according to XEP-0174.

- interface LLServiceConnectionListener

  Notification about when new Link-local connections associated with a specific Link-local service has been established.

- interface LLServiceListener

  Notification for new Link-local services created.

- interface LLServiceStateListener

  Interface for handeling link-local service events such as service closing, service crashes and other events.

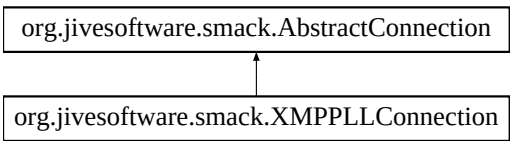
- class XMPPLLConnection

  Link-local XMPP connection according to XEP-0174 connection.

2 Class Documentation

2.1 org.jivesoftware.smack.AbstractConnection Class Reference

Inheritance diagram for org.jivesoftware.smack.AbstractConnection::



Public Member Functions

- String getConnectionID ()

  Returns the connection ID for this connection, which is the value set by the server when opening a XMPP stream.

- String getServiceName ()

  Returns the name of the service provided by the XMPP server for this connection.

- String getHost ()

  Returns the host name of the server where the XMPP server is running.

- int getPort ()

  Returns the port number of the XMPP server for this connection.

- abstract String getUser ()

  Returns the full XMPP address of the user that of the connection.

- boolean isConnected ()

  Returns true if currently connected to the XMPP server.

- void sendPacket (Packet packet)

  Sends the specified packet to the server.

- void addPacketListener (PacketListener packetListener, PacketFilter packetFilter)

  Registers a packet listener with this connection.

- void removePacketListener (PacketListener packetListener)

  Removes a packet listener from this connection.

- void addPacketWriterListener (PacketListener packetListener, PacketFilter packetFilter)

  Registers a packet listener with this connection.

- void removePacketWriterListener (PacketListener packetListener)

  Removes a packet listener from this connection.

- void addPacketWriterInterceptor (PacketInterceptor packetInterceptor, PacketFilter packetFilter)

  Registers a packet interceptor with this connection.

- void removePacketWriterInterceptor (PacketInterceptor packetInterceptor)

  Removes a packet interceptor.

- PacketCollector createPacketCollector (PacketFilter packetFilter)

  Creates a new packet collector for this connection.

- void addConnectionListener (ConnectionListener connectionListener)

  Adds a connection listener to this connection that will be notified when the connection closes or fails.

- void removeConnectionListener (ConnectionListener connectionListener)

  Removes a connection listener from this connection.

Static Public Attributes

- static boolean DEBUG_ENABLED = true

  Value that indicates whether debugging is enabled.

Protected Member Functions

- abstract void shutdown ()

  Closes the connection by closing the stream.

- abstract void disconnect ()

  Closes the connection.

Protected Attributes

- SmackDebugger **debugger** = null
- String host

  IP address or host name of the server.

- int **port**
- Socket **socket**
- boolean **connected** = false
- Writer **writer**
- Reader **reader**
- PacketWriter **packetWriter**
- AbstractPacketReader **packetReader**
- String serviceName

  Service name of the XMPP connection.

Static Package Functions

- **[static initializer]**

Package Attributes

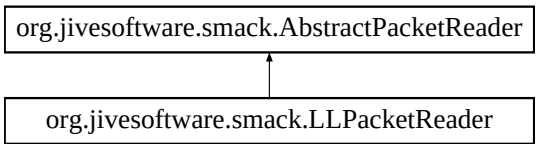
- int **connectionCounterValue** = connectionCounter.getAndIncrement()
- String **connectionID** = null

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smack/AbstractConnection.java

2.2 org.jivesoftware.smack.AbstractPacketReader Class Reference

Listens for XML traffic from the XMPP server and parses it into packet objects. Inheritance diagram for org.jivesoftware.smack.AbstractPacketReader::



Classes

- class **ListenerNotification**

  A runnable to notify all listeners of a packet.

- class **ListenerWrapper**

  A wrapper class to associate a packet filter with a listener.

Public Member Functions

- PacketCollector createPacketCollector (PacketFilter packetFilter)

  Creates a new packet collector for this reader.

- void addPacketListener (PacketListener packetListener, PacketFilter packetFilter)

  Registers a packet listener with this reader.

- void removePacketListener (PacketListener packetListener)

  Removes a packet listener.

- void startup () throws XMPPException

  Starts the packet reader thread and returns once a connection to the server has been established.

- void shutdown ()

  Shuts the packet reader down.

Protected Member Functions

- **AbstractPacketReader** (final AbstractConnection connection)
- void init ()

  Initializes the reader in order to be used.

- void **cancelPacketCollector** (PacketCollector packetCollector)
- abstract void parsePackets (Thread thread)

  Parse top-level packets in order to process them further.

- void notifyReconnection ()

  Sends a notification indicating that the connection was reconnected successfully.

- void resetParser ()

  Resets the parser using the latest connection's reader.

- void releaseConnectionIDLock ()

  Releases the connection ID lock so that the thread that was waiting can resume.

- void processPacket (Packet packet)

  Processes a packet after it's been fully parsed by looping through the installed packet collectors and listeners and letting them examine the packet to see if they are a match with the filter.

- StreamError **parseStreamError** (XmlPullParser parser) throws IOException, XmlPullParserException

Protected Attributes

- Thread **readerThread**
- XmlPullParser **parser**
- boolean **done**
- final Map< PacketListener, ListenerWrapper > **listeners**
- final Collection< ConnectionListener > **connectionListeners**
- String **connectionID** = null

Package Functions

- void cleanup ()

  Cleans up all resources used by the packet reader.

- void notifyConnectionError (Exception e)

  Sends out a notification that there was an error with the connection and closes the connection.

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smack/AbstractPacketReader.java

2.3 org.jivesoftware.smackx.packet.CapsExtension Class Reference

Inherits org::jivesoftware::smack::packet::PacketExtension.

Public Member Functions

- **CapsExtension** (String node, String version, String hash)
- String **getElementName** ()
- String **getNamespace** ()
- String **getNode** ()
- void **setNode** (String node)
- String **getVersion** ()
- void **setVersion** (String version)
- String **getHash** ()
- void **setHash** (String hash)
- String **toXML** ()

Static Public Attributes

- static final String **XMLNS** = "http://jabber.org/protocol/caps"
- static final String **NODE_NAME** = "c"

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smackx/packet/CapsExtension.java

2.4 org.jivesoftware.smackx.provider.CapsExtensionProvider Class Reference

Inherits org::jivesoftware::smack::provider::PacketExtensionProvider.

Public Member Functions

- PacketExtension **parseExtension** (XmlPullParser parser) throws Exception

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smackx/provider/CapsExtensionProvider.java

2.5 org.jivesoftware.smackx.EntityCapsManager.CapsPacketListener Class Reference

Inherits org::jivesoftware::smack::PacketListener.

Public Member Functions

- void **processPacket** (Packet packet)

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smackx/EntityCapsManager.java

2.6 org.jivesoftware.smackx.CapsVerListener Interface Reference

Inherited by org.jivesoftware.smackx.LLServiceDiscoveryManager.CapsPresenceRenewer.

Public Member Functions

- void **capsVerUpdated** (String capsVer)

The documentation for this interface was generated from the following file:

- source/org/jivesoftware/smackx/CapsVerListener.java

2.7 org.jivesoftware.smack.LLService.CollectorWrapper Class Reference

Packet Collector Wrapper which is used for collecting packages from multiple connections as well as newly established connections (works together with LLService constructor.

Public Member Functions

- synchronized Packet nextResult (long timeout)

  Returns the next available packet.

- void **cancel** ()

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smack/LLService.java

2.8 org.jivesoftware.smackx.EntityCapsManager Class Reference

Keeps track of entity capabilities.

Classes

- class CapsPacketListener

Public Member Functions

- void addUserCapsNode (String user, String node)

  Add a record telling what entity caps node a user has.

- void removeUserCapsNode (String user)

Remove a record telling what entity caps node a user has.

- String getNodeVersionByUser (String user)
  Get the Node version (node.ver) of a user.
- DiscoverInfo getDiscoverInfoByUser (String user)
  Get the discover info given a user name.
- String getCapsVersion ()
  Get our own caps version.
- String getNode ()
  Get our own entity node.
- void setNode (String node)
  Set our own entity node.
- void **addPacketListener** (AbstractConnection connection)
- void **addCapsVerListener** (CapsVerListener listener)
- void **removeCapsVerListener** (CapsVerListener listener)
- void **spam** ()
- void setCurrentCapsVersion (DiscoverInfo discoverInfo, String capsVersion)
  Set our own caps version.

Static Public Member Functions

- static void addDiscoverInfoByNode (String node, DiscoverInfo info)
  Add DiscoverInfo to the database.
- static DiscoverInfo getDiscoverInfoByNode (String node)
  Retrieve DiscoverInfo for a specific node.

Static Public Attributes

- static final String **HASH_METHOD** = "sha-1"
- static final String **HASH_METHOD_CAPS** = "SHA-1"

Package Functions

- void **calculateEntityCapsVersion** (DiscoverInfo discoverInfo, String identityType, String identityName, List< String > features, DataForm extendedInfo)

Static Package Functions

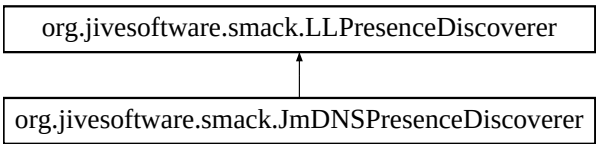
- **[static initializer]**

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smackx/EntityCapsManager.java

2.9 org.jivesoftware.smack.JmDNSPresenceDiscoverer Class Reference

An implementation of LLPresenceDiscoverer using JmDNS. Inheritance diagram for org.jivesoftware.smack.JmDNSPresenceDiscoverer::



Classes

- class **PresenceServiceListener**

  Implementation of a JmDNS ServiceListener.

Static Protected Attributes

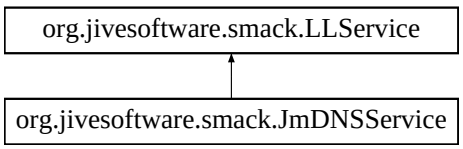
- static final int **SERVICE_REQUEST_TIMEOUT** = 10000
- static JmDNS **jmdns**

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smack/JmDNSPresenceDiscoverer.java

2.10 org.jivesoftware.smack.JmDNSService Class Reference

Implements a LLService using JmDNS. Inheritance diagram for org.jivesoftware.smack.JmDNSService::



Public Member Functions

- void **close** ()
- void **serviceNameChanged** (String newName, String oldName)
- void makeUnavailable ()

  Unregister the DNS-SD service, making the client unavailable.

- void spam ()

  Spam stdout with some debug information.

Static Public Member Functions

- static LLService create (LLPresence presence) throws XMPPException

  Instantiate a new JmDNSService and start to listen for connections.

- static LLService create (LLPresence presence, InetAddress addr) throws XMPPException

  Instantiate a new JmDNSService and start to listen for connections.

Protected Member Functions

- void updateText ()

  Update the text field information.

- void registerService () throws XMPPException

  Register the DNS-SD service with the daemon.

- void reannounceService () throws XMPPException

  Reregister the DNS-SD service with the daemon.

Static Package Attributes

- static JmDNS **jmdns** = null
- static JmDNSPresenceDiscoverer **presenceDiscoverer** = null
- static final String **SERVICE_TYPE** = "_presence._tcp.local."

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smack/JmDNSService.java

2.11 org.jivesoftware.smack.LLChat Class Reference

Keeps track of a chat session between two link-local clients.

Public Member Functions

- String getServiceName ()

  Get the service name of the remote client of this chat session.

- void sendMessage (Message message) throws XMPPException

  Send a message packet to the remote client.

- Message **generateMessage** (String text)
- void sendMessage (String text) throws XMPPException

  Send a message to the remote client.

- void addMessageListener (LLMessageListener listener)

  Add a message listener.

Package Functions

- **LLChat** (LLService service, LLPresence presence) throws XMPPException
- void deliver (Message message)

  Deliver a message to the message listeners.

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smack/LLChat.java

2.12 org.jivesoftware.smack.LLChatListener Interface Reference

Notification about new Link-local chat sessions.

Public Member Functions

- void newChat (LLChat chat)

  New chat has been created.

- void chatInvalidated (LLChat chat)

  Called when a chat session is invalidated (due to service name changes.

The documentation for this interface was generated from the following file:

- source/org/jivesoftware/smack/LLChatListener.java

2.13 org.jivesoftware.smack.LLConnectionConfiguration Class Reference

Link-local connection configuration settings.

Public Member Functions

- boolean isInitiator ()

  Tells if the connection is the initiating one.

- String getRemoteServiceName ()

  Return the service name of the remote peer.

- String getLocalServiceName ()

  Return the service name of this client.

- LLPresence getLocalPresence ()

  Return this clients link-local presence information.

- LLPresence getRemotePresence ()

  Return the remote client's link-local presence information.

- Socket getSocket ()

  Return the socket which has been established when the remote client connected.

Package Functions

- LLConnectionConfiguration (LLPresence local, LLPresence remote)

  Configuration used for connecting to remote peer.

- LLConnectionConfiguration (LLPresence local, Socket remoteSocket)

  Instantiate a link-local configuration when the connection is acting as the host.

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smack/LLConnectionConfiguration.java

2.14 org.jivesoftware.smack.LLConnectionListener Interface Reference

Notification about when new Link-local connections has been established.

Public Member Functions

- void connectionCreated (XMPPLLConnection connection)

  A new link-local connection has been established.

The documentation for this interface was generated from the following file:

- source/org/jivesoftware/smack/LLConnectionListener.java

2.15 org.jivesoftware.smack.LLMessageListener Interface Reference

Notification when messages are being delivered to a chat.

Package Functions

- void processMessage (LLChat chat, Message message)

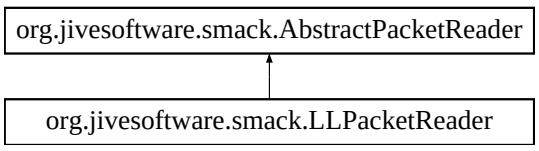
  New message in chat.

The documentation for this interface was generated from the following file:

- source/org/jivesoftware/smack/LLMessageListener.java

2.16 org.jivesoftware.smack.LLPacketReader Class Reference

Listens for XML traffic from a remote XMPP client and parses it into packet objects. Inheritance diagram for org.jivesoftware.smack.LLPacketReader::



Public Member Functions

- **LLPacketReader** (final LLService service, final XMPPLLConnection connection)

Protected Member Functions

- void parsePackets (Thread thread)

  Parse top-level packets in order to process them further.

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smack/LLPacketReader.java

2.17 org.jivesoftware.smack.LLPresence Class Reference

Class for describing a Link-local presence information according to XEP-0174.

Public Types

- enum **Mode** { **avail**, **away**, **dnd** }

Public Member Functions

- **LLPresence** (String serviceName)
- **LLPresence** (String serviceName, String host, int port)
- **LLPresence** (String serviceName, String host, int port, List< Tuple< String, String >> records)
- List< Tuple< String, String > > **toList** ()
- void **setServiceName** (String serviceName)
- void **setFirstName** (String name)
- void **setLastName** (String name)
- void **setEMail** (String email)
- void **setMsg** (String msg)
- void **setNick** (String nick)
- void **setStatus** (Mode status)
- void **setJID** (String jid)
- void **setHash** (String hash)
- void **setNode** (String node)
- void **setVer** (String ver)
- String **getFirstName** ()
- String **getLastName** ()
- String **getEMail** ()
- String **getMsg** ()
- String **getNick** ()
- Mode **getStatus** ()
- String **getJID** ()
- String **getServiceName** ()
- String **getHost** ()

- String **getHash** ()
- String **getNode** ()
- String **getVer** ()
- int **getPort** ()
- String **getValue** (String key)
- void **putValue** (String key, String value)
- boolean **equals** (Object o)
- int **hashCode** ()

Package Functions

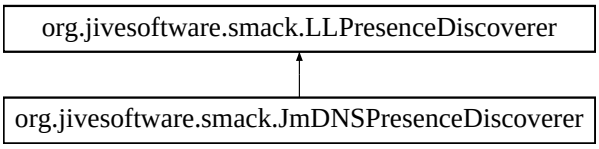
- void update (LLPresence p)
  Update all the values of the presence.
- void **setPort** (int port)

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smack/LLPresence.java

2.18 org.jivesoftware.smack.LLPresenceDiscoverer Class Reference

Link-local presence discoverer. Inheritance diagram for org.jivesoftware.smack.LLPresenceDiscoverer::



Public Member Functions

- void addPresenceListener (LLPresenceListener listener)
  Add listener which will be notified when new presences are discovered, presence information changed or presences goes offline.
- void removePresenceListener (LLPresenceListener listener)
  Remove presence listener.
- Collection< LLPresence > getPresences ()
  Return a collection of presences known.
- LLPresence getPresence (String name)
  Return the presence with the specified service name.

Protected Member Functions

- void presenceAdded (String name)
  Used by the class extending this one to tell when new presence is added.
- void presenceInfoAdded (String name, LLPresence presence)
  Used by the class extending this one to tell when new presence information is added.
- void presenceRemoved (String name)
  Used by the class extending htis one to tell when a presence goes offline.

Protected Attributes

- Set< LLPresenceListener > **listeners** = new CopyOnWriteArraySet<LLPresenceListener>()

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smack/LLPresenceDiscoverer.java

2.19 org.jivesoftware.smack.LLPresenceListener Interface Reference

Interface for receiving notifications about presence changes.

Public Member Functions

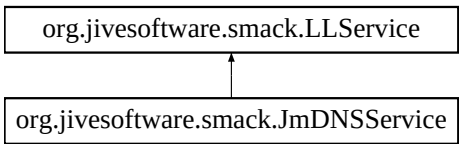
- void presenceNew (LLPresence presence)
  New link-local presence has been discovered.
- void presenceRemove (LLPresence presence)
  A link-local presence has gone offline.

The documentation for this interface was generated from the following file:

- source/org/jivesoftware/smack/LLPresenceListener.java

2.20 org.jivesoftware.smack.LLService Class Reference

LLService acts as an abstract interface to a Link-local XMPP service according to XEP-0174. Inheritance diagram for org.jivesoftware.smack.LLService::



Classes

- class CollectorWrapper

  Packet Collector Wrapper which is used for collecting packages from multiple connections as well as newly established connections (works together with LLService constructor.

- class **ConnectionActivityListener**

  ConnectionActivityListener listens for link-local connection activity such as closed connection and broken connection, and keeps record of what active connections exist up to date.

- class **ConnectionInitiatorThread**

  Initiates a connection in a seperate thread, controlling it was established correctly and stream was initiated.

- class **ListenerWrapper**

  A wrapper class to associate a packet filter with a listener.

- class **MessageListener**

  MessageListener listenes for messages from connections and delivers them to the corresponding chat session.

Public Member Functions

- void spam ()

  Spam stdout with some debug information.

- abstract void makeUnavailable ()

  Make the client unavailabe.

- void **init** () throws XMPPException
- void **close** ()
- void addLLServiceConnectionListener (LLServiceConnectionListener listener)

  Adds a listener that are notified when a new link-local connection has been established.

- void removeLLServiceConnectionListener (LLServiceConnectionListener listener)

Removes a listener from the new connection listener list.

- void addPacketListener (PacketListener listener, PacketFilter filter)

  Add a packet listener.

- void removePacketListener (PacketListener listener)

  Remove a packet listener.

- void addServiceStateListener (LLServiceStateListener listener)

  Add service state listener.

- void removeServiceStateListener (LLServiceStateListener listener)

  Remove service state listener.

- void addLLChatListener (LLChatListener listener)

  Add Link-local chat session listener.

- void removeLLChatListener (LLChatListener listener)

  Remove Link-local chat session listener.

- void addPresenceListener (LLPresenceListener listener)

  Add presence listener.

- void removePresenceListener (LLPresenceListener listener)

  Remove presence listener.

- LLPresence getPresenceByServiceName (String serviceName)

  Get the presence information associated with the given service name.

- CollectorWrapper **createPacketCollector** (PacketFilter filter)
- Collection< XMPPLLConnection > getConnections ()

  Return a collection of all active connections.

- LLChat getChat (String serviceName) throws XMPPException

  Get a LLChat associated with a given service name.

- XMPPLLConnection getConnection (String serviceName) throws XMPPException

  Returns a XMPPLLConnection to the serviceName.

- void sendPacket (Packet packet) throws XMPPException

  Send a packet to the remote peer.

- IQ getIQResponse (IQ request) throws XMPPException

  Send an IQ set or get and wait for the response.

- void updatePresence (LLPresence presence) throws XMPPException

  Update the presence information announced by the mDNS/DNS-SD daemon.

- LLPresence getLocalPresence ()

  Get current Link-local presence.

Static Public Member Functions

- static void addLLServiceListener (LLServiceListener listener)

  Add a LLServiceListener.

- static void removeLLServiceListener (LLServiceListener listener)

  Remove a LLServiceListener.

- static void notifyServiceListeners (LLService service)

  Notify LLServiceListeners about a new Link-local service.

Protected Member Functions

- **LLService** (LLPresence presence, LLPresenceDiscoverer discoverer)
- abstract void registerService () throws XMPPException

  Returns the running mDNS/DNS-SD XMPP instance.

- abstract void reannounceService () throws XMPPException

  Re-announce the presence information by using the mDNS/DNS-SD daemon.

- abstract void updateText ()

  Update the text field information.

- void **unknownOriginMessage** (Message message)
- void **serviceNameChanged** (String newName, String oldName)

Protected Attributes

- LLPresence **presence**

Package Functions

- XMPPLLConnection getConnectionTo (String serviceName)

  Returns a connection to a given service name.

- void **addIngoingConnection** (XMPPLLConnection connection)
- void **removeIngoingConnection** (XMPPLLConnection connection)

- void **addOutgoingConnection** (XMPPLLConnection connection)
- void **removeOutgoingConnection** (XMPPLLConnection connection)
- LLChat **removeLLChat** (String serviceName)
- void **newLLChat** (LLChat chat)
- void sendMessage (Message message) throws XMPPException

  Send a message to the remote peer.

Static Package Functions

- **[static initializer]**

Static Package Attributes

- static final int **DEFAULT_MIN_PORT** = 2300
- static final int **DEFAULT_MAX_PORT** = 2400

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smack/LLService.java

2.21 org.jivesoftware.smack.LLServiceConnectionListener Interface Reference

Notification about when new Link-local connections associated with a specific Link-local service has been established.

Inherited by org.jivesoftware.smackx.LLServiceDiscoveryManager.ConnectionServiceMaintainer.

Public Member Functions

- void connectionCreated (XMPPLLConnection connection)

  A new link-local connection has been established.

The documentation for this interface was generated from the following file:

- source/org/jivesoftware/smack/LLServiceConnectionListener.java

2.22 org.jivesoftware.smackx.LLServiceDiscoveryManager Class Reference

LLServiceDiscoveryManager acts as a wrapper around ServiceDiscoveryManager as ServiceDiscoveryManager only creates an interface for requesting service information on existing connections.

Classes

- class **CapsPresenceRenewer**
- class **ConnectionServiceMaintainer**

  In case that a connection is unavailable we create a new connection and push the service discovery procedure until the new connection is established.

Public Member Functions

- void addDiscoverInfoTo (DiscoverInfo response)

  Add discover info response data.

- DiscoverInfo getOwnDiscoverInfo ()

  Get a DiscoverInfo for the current entity caps node.

- void setExtendedInfo (DataForm info)

  Registers extended discovery information of this XMPP entity.

- void removeExtendedInfo ()

  Removes the dataform containing extended service discovery information from the information returned by this XMPP entity.

- DiscoverInfo discoverInfo (String serviceName) throws XMPPException

  Returns the discovered information of a given XMPP entity addressed by its JID.

- DiscoverInfo discoverInfo (String serviceName, String node) throws XMPPException

  Returns the discovered information of a given XMPP entity addressed by its JID and note attribute.

- DiscoverItems discoverItems (String entityID) throws XMPPException

  Returns the discovered items of a given XMPP entity addressed by its JID.

- DiscoverItems discoverItems (String serviceName, String node) throws XMPPException

  Returns the discovered items of a given XMPP entity addressed by its JID and note attribute.

- void setNodeInformationProvider (String node, NodeInformationProvider listener) throws XMPPException

  Sets the NodeInformationProvider responsible for providing information (ie items) related to a given node.

- void removeNodeInformationProvider (String node) throws XMPPException

  Removes the NodeInformationProvider responsible for providing information (ie items) related to a given node.

- Iterator< String > getFeatures ()

  Returns the supported features by this XMPP entity.

- void addFeature (String feature)

  Registers that a new feature is supported by this XMPP entity.

- void removeFeature (String feature) throws XMPPException

  Removes the specified feature from the supported features by this XMPP entity.

- boolean includesFeature (String feature)

  Returns true if the specified feature is registered in the ServiceDiscoveryManager.

- boolean canPublishItems (String entityID) throws XMPPException

  Returns true if the server supports publishing of items.

- void publishItems (String entityID, DiscoverItems discoverItems) throws XMPPException

  Publishes new items to a parent entity.

- void publishItems (String entityID, String node, DiscoverItems discoverItems) throws XMPPException

  Publishes new items to a parent entity and node.

Static Public Member Functions

- static LLServiceDiscoveryManager getInstanceFor (LLService service)

  Get the LLServiceDiscoveryManager instance for a specific Link-local service.

- static String getIdentityName ()

  Returns the name of the client that will be returned when asked for the client identity in a disco request.

- static void setIdentityName (String name)

  Sets the name of the client that will be returned when asked for the client identity in a disco request.

- static String getIdentityType ()

  Returns the type of client that will be returned when asked for the client identity in a disco request.

- static void setIdentityType (String type)

  Sets the type of client that will be returned when asked for the client identity in a disco request.

Static Package Functions

- **[static initializer]**

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smackx/LLServiceDiscoveryManager.java

2.23 org.jivesoftware.smack.LLServiceListener Interface Reference

Notification for new Link-local services created.

Public Member Functions

- void serviceCreated (LLService service)
  The function called when a new Link-local service is created.

The documentation for this interface was generated from the following file:

- source/org/jivesoftware/smack/LLServiceListener.java

2.24 org.jivesoftware.smack.LLServiceStateListener Interface Reference

Interface for handeling link-local service events such as service closing, service crashes and other events.

Public Member Functions

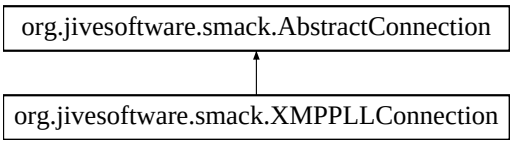
- void serviceNameChanged (String newName, String oldName)
  Notification that the service name was changed.
- void serviceClosed ()
  Notification that the connection was closed normally.
- void serviceClosedOnError (Exception e)
  Notification that the connection was closed due to an exception.
- void unknownOriginMessage (Message e)
  Notification that a message with unknown presence was received.

The documentation for this interface was generated from the following file:

- source/org/jivesoftware/smack/LLServiceStateListener.java

2.25 org.jivesoftware.smack.XMPPLLConnection Class Reference

Link-local XMPP connection according to XEP-0174 connection. Inheritance diagram for org.jivesoftware.smack.XMPPLLConnection::



Public Member Functions

- boolean isInitiator ()

  Tells if this connection instance is the initiator.

- String getUser ()

  Return the user name of the remote peer (service name).

- void setServiceName (String serviceName)

  Sets the name of the service provided in the <stream:stream .

- void streamInitiatingReceived () throws XMPPException

  Handles the opening of a stream after a remote client has connected and opened a stream.

- void sendPacket (Packet packet)

  Sends the specified packet to the remote peer.

- void disconnect ()

  Closes the connection.

Static Public Member Functions

- static void addLLConnectionListener (LLConnectionListener listener)

  Adds a listener that are notified when a new link-local connection has been established.

- static void removeLLConnectionListener (LLConnectionListener listener)

  Removes a listener from the new connection listener list.

Protected Member Functions

- void shutdown ()
  Closes the connection by closing the stream.

Protected Attributes

- XMPPLLConnection **connection**

Package Functions

- XMPPLLConnection (LLService service, LLConnectionConfiguration config)
  Instantiate a new link-local connection.
- void setRemotePresence (LLPresence remotePresence)
  Set the remote presence.
- void initListen () throws XMPPException
  Start listen for data and a stream tag.
- void connect () throws XMPPException
  Create a socket, connect to the remote peer and initiate a XMPP stream session.
- void updateLastActivity ()
  Update the timer telling when the last activity happend.

The documentation for this class was generated from the following file:

- source/org/jivesoftware/smack/XMPPLLConnection.java

F Code Documentation: Space Client

Space Client

Generated by Doxygen 1.6.1

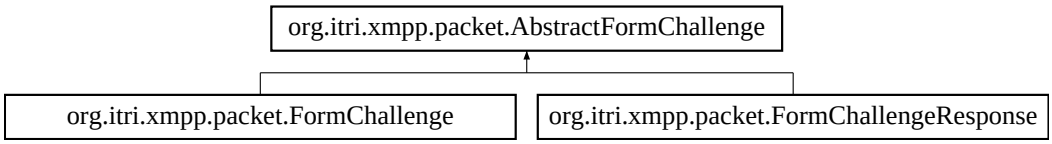
Sun Dec 6 15:13:06 2009

Contents

1 Class Documentation

1.1 org.itri.xmpp.packet.AbstractFormChallenge Class Reference

An abstract form challange class providing basic functionality for form challenges and responses. Inheritance diagram for org.itri.xmpp.packet.AbstractFormChallenge::



Public Member Functions

- abstract String getElementName ()

  Get the element name.

- String getNamespace ()

  Get the namespace.

- Form getChallengeForm ()

  Get the form.

- String toXML ()

  Convert to an XML String.

Static Public Attributes

- static String **XMLNS** = "urn:itri:xmpp:space:challenge:form"

Protected Member Functions

- **AbstractFormChallenge** (Form form)

Protected Attributes

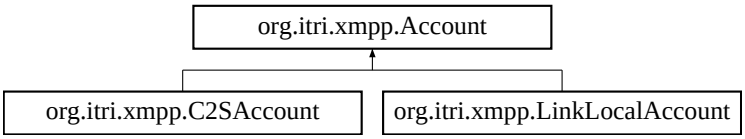
- Form **challenge**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/packet/AbstractFormChallenge.java

1.2 org.itri.xmpp.Account Class Reference

Account class providing basic functionality common for all types of accounts. Inheritance diagram for org.itri.xmpp.Account::



Public Member Functions

- abstract String getType ()

  Get a human readable name of the account type.

- String getAccountName ()

  Return the name of the account.

- Session getActiveSession ()

  Return the active session associated with this account.

- String **toString** ()

Protected Member Functions

- **Account** (String accountName)

Package Functions

- void setAccountName (String accountName)

  Set the name of the account.

- void setActiveSession (Session session)
  Set the active session of this account.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/Account.java

1.3 org.itri.xmpp.AccountManager Class Reference

Manages accounts.

Public Member Functions

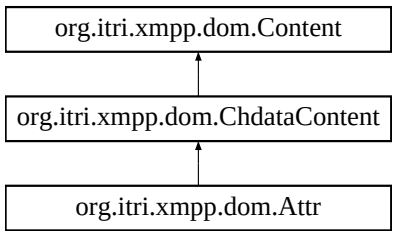
- void addAccount (Account account) throws ClientException
  Add an account to the account manager.
- Account getAccount (String accountName)
  Get the account given the account name.
- Collection< Account > getAccounts ()
  Return a collection of all the accounts.
- void loadXMLConfig (String filename) throws ClientException
  Load account accounts from a XML file.
- void saveXMLConfig (String filename)
  Save accounts to an XML file.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/AccountManager.java

1.4 org.itri.xmpp.dom.Attr Class Reference

XML Attribute document node. Inheritance diagram for org.itri.xmpp.dom.Attr::



Public Member Functions

- Attr (String rid, String ns, String name, String chdata)
  Create a new attribute.
- Attr (String rid, String name, String chdata)
  Create a new attribute.
- void setName (String newName)
  Set the name of this attribute.
- String getName ()
  Get the name of this attribute.
- void setPrefix (String newNs)
  Set the namespace prefix of this attribute.
- String getPrefix ()
  Get the namespace prefix of this attribute.
- String toXML ()
  Generate the XML code for this attribute.

Protected Member Functions

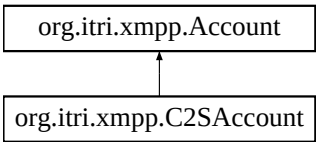
- Content _cloneContent ()
  Clone attribute.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/dom/Attr.java

1.5 org.itri.xmpp.C2SAccount Class Reference

Inheritance diagram for org.itri.xmpp.C2SAccount::



Public Member Functions

- **C2SAccount** (String accountName)
- String getType ()

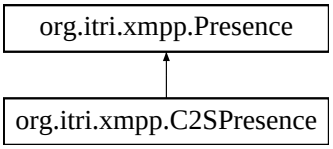
  Get a human readable name of the account type.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/C2SAccount.java

1.6 org.itri.xmpp.C2SPresence Class Reference

Regular XMPP C2S presence. Inheritance diagram for org.itri.xmpp.C2SPresence::



Public Member Functions

- **C2SPresence** (String entity)
- void setContact (Contact contact)

  Set what contact this presence is associated with.

- Contact getContact ()

  Specify what contact this presence is associated with.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/C2SPresence.java

1.7 org.itri.xmpp.C2SSession Class Reference

Inherits org::itri::xmpp::Session< C2SAccount, C2SPresence >.

Public Member Functions

- **C2SSession** (C2SAccount account, C2SPresence presence)
- synchronized void notifyAboutAllContacts (ContactListener listener)

  Notify contact listener about all known contacts.

- void addContactListener (ContactListener contactListener)

  Add a contact listener.

- synchronized void addContactListenerAndGetAll (ContactListener contactListener)

  Add contact listener and retrieve all items.

- void removeContactListener (ContactListener contactListener)

  Remove a contact listener.

- void addPacketListener (PacketListener listener, PacketFilter filter)

  Add a packet listener with a given filter.

- void removePacketListener (PacketListener listener)

  Remove a packet listener.

- void **updatePresence** (Presence presence)
- void **updatePresence** (Presence.Mode mode, String message)
- void **updatePresence** (Presence.Mode mode)
- void **updatePresence** (String message)
- Chat **newChat** (String entity)
- void sendPacket (Packet packet) throws ClientException

  Send a packet.

- IQ getIQResponse (IQ iq) throws ClientException

  Perform a IQ request and get the response.

- DiscoverInfo discoverInfo (String entity) throws ClientException

  Discover info on the given entity.

- void addFeature (String feature)

  Add Service Discovery feature to this session.

- void **spam** ()

Protected Member Functions

- synchronized void contactAdded (Contact contact)

  Called when a new contact has been added.

- synchronized void contactRemoved (Contact contact)

  Called when a contact has been removed.

Protected Attributes

- Map< String, Contact > **contacts**

Package Functions

- void start () throws ClientException

  Start the session.

- void stop () throws ClientException

  Stop the session.

- boolean contactExists (String name)

  Returns ture if the contact associated with the name exists.

- Contact getContact (String name)

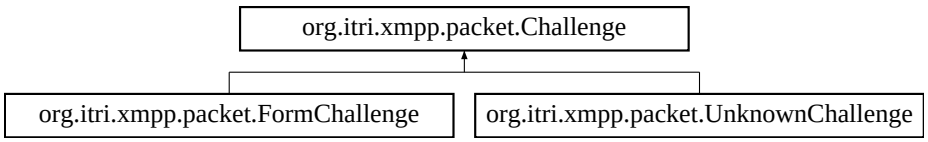
  Get the contact as a Contact class associated with a name.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/C2SSession.java

1.8 org.itri.xmpp.packet.Challenge Interface Reference

The interface for challenges. Inheritance diagram for org.itri.xmpp.packet.Challenge::

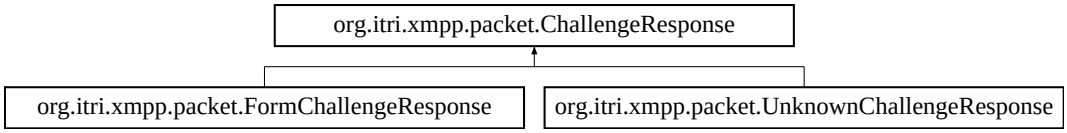


The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/packet/Challenge.java

1.9 org.itri.xmpp.packet.ChallengeResponse Interface Reference

The interface for challenge responses. Inheritance diagram for org.itri.xmpp.packet.ChallengeResponse::



The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/packet/ChallengeResponse.java

1.10 org.itri.xmpp.Chat< SessionType extends Session > Class Reference

Class for managing instant messaging chats (participant and message history).

Public Member Functions

- void addChatMessageListener (ChatMessageListener listener)
  Add chat message listener.
- void removeChatMessageListener (ChatMessageListener listener)
  Remove chat message listener.
- void deliver (Message m)
  Deliver message to the chat (remote participant).
- SessionType getSession ()
  Get the associated session.
- List< Message > getHistory ()
  Get the chat history.
- abstract void sendMessage (String text) throws ClientException
  Send a message to the remote participant.
- abstract String getParticipant ()
  Get the service name/JID of the remote participant.

Protected Member Functions

- Chat (SessionType session)
  Instantiate a new chat associated with the given session.
- void deliverOwnMessage (Message m)

Deliver own message.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/Chat.java

1.11 org.itri.xmpp.ChatManager Class Reference

Class used for managing all chats of all sessions of a session manager.

Public Member Functions

- void addNewChatListener (NewChatListener listener)
  Add new chat listener.
- void removeNewChatListener (NewChatListener listener)
  Remove chat listener.
- Set< Chat > getAllChats ()
  Get all active chat sessions.
- void sessionActivating (Session session)
  Called when a session is activating.
- void sessionActivated (Session session)
  Called when a session has finished activating.
- void sessionDeactivated (Session session)
  Called when a session has been deactivated.
- void sessionActivationFailed (Session session, ClientException ce)
  Called when a session activation failed due to an exception.

Package Functions

- ChatManager (SessionManager sm)
  Create a new chat manager.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ChatManager.java

1.12 org.itri.xmpp.ChatMessageListener Interface Reference

Interface for receiving chat messages.

Public Member Functions

- void newMessage (Message m)

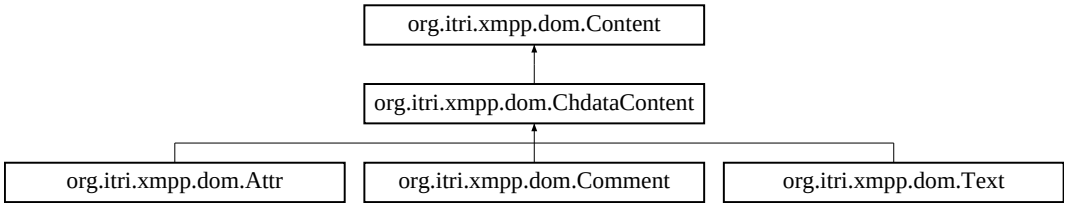
  A new message has been received.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/ChatMessageListener.java

1.13 org.itri.xmpp.dom.ChdataContent Class Reference

An abstract class for XML nodes with Character data. Inheritance diagram for org.itri.xmpp.dom.ChdataContent::



Public Member Functions

- void setChdata (String newChdata)

  Set the character data of this node.

- String getChdata ()

  Get the character data of this node.

Protected Member Functions

- ChdataContent (String rid)

  Create a new abstract character data node.

Protected Attributes

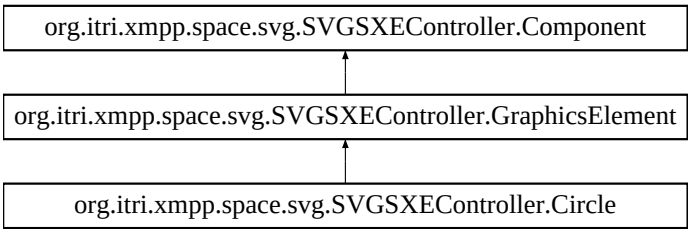
- String **chdata** = null

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/dom/ChdataContent.java

1.14 org.itri.xmpp.space.svg.SVGSXEController.Circle Class Reference

SVG Circle. Inheritance diagram for org.itri.xmpp.space.svg.SVGSXEController.Circle::



Public Member Functions

- **Circle** (Element e) throws ClientException
- **Circle** (float x, float y, float r, SVGPaint p)
- float **getX** ()
- float **getY** ()
- float **getR** ()
- void move (float x, float y) throws ClientException

  Move the circle.

- Element **generateShapeElement** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/svg/SVGSXEController.java

1.15 org.itri.xmpp.ClientException Class Reference

An exception caused by something in org.itri.xmpp;.

Public Member Functions

- ClientException (String text)

  A client exception with a descriptive message describing what caused the error.

- ClientException (Exception e)

  A client exception with bundled with an exception causing the error.

- ClientException (String text, Exception e)

  A client exception bundled with the causing exception as well as a descriptive message.

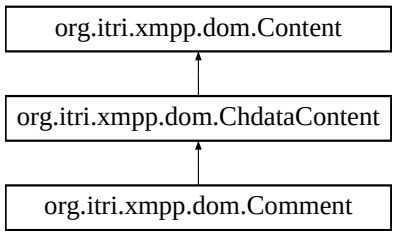
- void **printStackTrace** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ClientException.java

1.16 org.itri.xmpp.dom.Comment Class Reference

XML Comment DOM node. Inheritance diagram for org.itri.xmpp.dom.Comment::



Public Member Functions

- Comment (String rid, String chdata)

  Create a new XML comment.

- String toXML ()

  Generate XML code for this XML comment.

Protected Member Functions

- Content _cloneContent ()

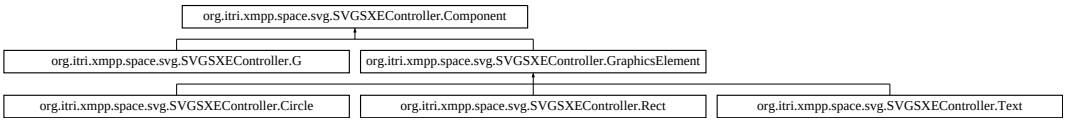
  Clone comment.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/dom/Comment.java

1.17 org.itri.xmpp.space.svg.SVGSXEController.Component Class Reference

Abstract class for a SVG tree component. Inheritance diagram for org.itri.xmpp.space.svg.SVGSXEController.Component::



Public Member Functions

- **Component** (Element e)
- Element **getElement** ()
- String **getRid** ()

Protected Member Functions

- abstract Element **generateElement** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/svg/SVGSXEController.java

1.18 org.itri.xmpp.Configuration Class Reference

Read the client configuration XML file.

Public Member Functions

- Set< Account > getAccounts ()

  Return the accounts that was read from a configuration file.

Package Functions

- void load (String filename) throws ClientException

  Parse an XML file containing configuration.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/Configuration.java

1.19 org.itri.xmpp.Contact Class Reference

An abstract class providing basic functionality for all kinds of contacts.

Public Member Functions

- **Contact** (Session session, String user)
- String getUser ()

  Return the user name of the contact.

- String **toString** ()

Protected Attributes

- Session **session**
- String **user**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/Contact.java

1.20 org.itri.xmpp.ContactListener< ContactType extends Contact > Interface Reference

Notificate about contacts being added and removed.

Public Member Functions

- void contactAdded (ContactType contact)

  A contact was added to the session.

- void contactRemoved (ContactType concat)

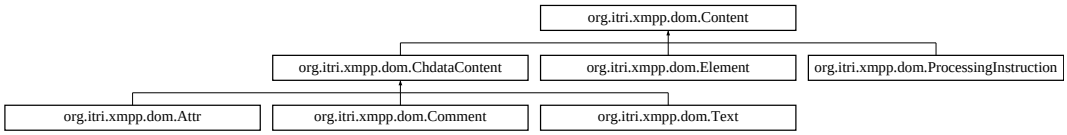
  A contact was removed.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/ContactListener.java

1.21 org.itri.xmpp.dom.Content Class Reference

General content of an XML DOM node. Inheritance diagram for org.itri.xmpp.dom.Content::



Public Member Functions

- String getRid ()

  Get the ID of this content.

- void setParent (Element parent)

  Set the parent element of this content.

- Element getParent ()

  Get the parent element of this content.

- float getPrimaryWeight ()

  Geth the primary weight of this content.

- void setPrimaryWeight (float primaryWeight)

  Set the primary weight of this content.

- int getVersion ()

  Get the version of this content.

- void setVersion (int version)

  Set the version of this content.

- void increaseVersion ()

  Increase the version of this content by 1.

- abstract String toXML ()

  Abstract function generating the XML code for this content.

- Content cloneContent ()

  Deep clone this content.

Protected Member Functions

- Content (String rid)
  Create a new Content.
- String cloneOrNull (String s)
  Decrease the version of this content by 1.
- void copyTo (Content c)
  Copy version and weight information to a given content.
- abstract Content _cloneContent ()
  Deep clone this content.

Protected Attributes

- String **rid**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/dom/Content.java

1.22 org.itri.xmpp.dom.Document Class Reference

A class representing the XML Document Object Model.

Public Member Functions

- Document ()
  Create a new DOM.
- Document (Content content)
  Create a new DOM with an initial content.
- Document (Collection< Content > contents)
  Create a new DOm with an intitial collection of contents.
- void addContent (Content content)
  Add a content to this document.
- Vector< Content > getAllContents ()
  Get all contents.
- String toXML ()

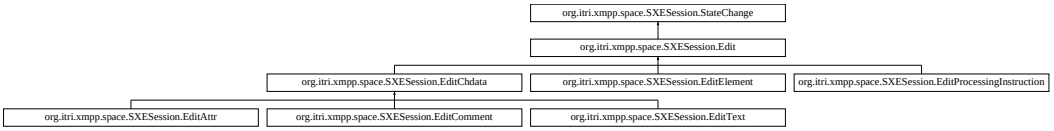
Generate XML Code of this document.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/dom/Document.java

1.23 org.itri.xmpp.space.SXESession.Edit Class Reference

State change that changes an existing node. Inheritance diagram for org.itri.xmpp.space.SXESession.Edit::



Public Member Functions

- int getVersion ()
  Get the version.
- String getTarget ()
  Get the target ID.
- abstract SXE.StateChange getStateChange ()
  Get the SXE instruction packet for this state change.

Protected Member Functions

- Edit (Content content, int version)
  Create a new state change.
- boolean needReinsert ()
  Return true if a reinsert is needed, in other words, if the primary weight or parent has been changed.
- void applyCommon () throws OrphanedContent
  Apply common changes.
- void revertCommon () throws OrphanedContent
  Revert common changes.
- void setCommonAttributes (SXE.Set set)
  Set common attributes for the set instruction.

Protected Attributes

- int **version**
- float **newPrimaryWeight** = Float.MIN_VALUE

Package Functions

- void setPrimaryWeight (float primaryWeight)

  Change the primary weight in this state change.

- void setParent (String parent)

  Change the parent in this state change.

- abstract void apply () throws OrphanedContent

  Apply this state change.

- abstract void revert () throws OrphanedContent

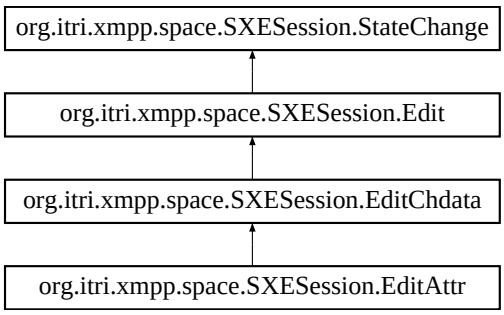
  Revert the content.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SXESession.java

1.24 org.itri.xmpp.space.SXESession.EditAttr Class Reference

State change for changing an attribute node. Inheritance diagram for org.itri.xmpp.space.SXESession.EditAttr::



Public Member Functions

- SXE.StateChange getStateChange ()

  Get the corresponding state change instruction.

Package Functions

- EditAttr (Attr attr, int version)

  Create a new state change for changing an attribute.

- void setName (String name)

  Change the name of the attribute.

- void setNs (String ns)

  Change the namespace prefix of the attribute.

- void apply () throws OrphanedContent

  Apply changes to the attribute.

- void revert () throws OrphanedContent

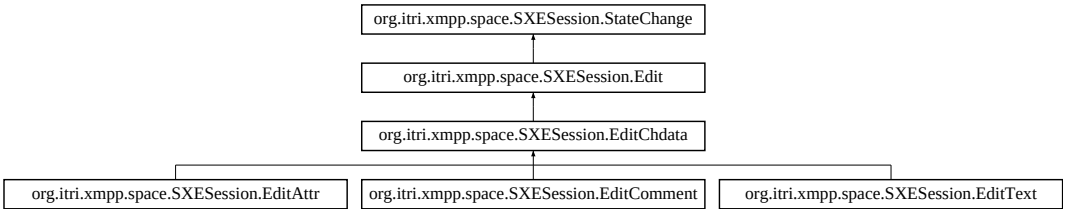
  Revert the changes of the attribute to it's previous state.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SXESession.java

1.25 org.itri.xmpp.space.SXESession.EditChdata Class Reference

State change that changes an existing node that contains character data. Inheritance diagram for org.itri.xmpp.space.SXESession.EditChdata::



Public Member Functions

- void setReplace (int replacefrom, int replacen)

  Make this state change so the new character data replaces a part of the old character data.

Protected Member Functions

- EditChdata (ChdataContent chcontent, int version)

  Create a new state change.

- void applyChdata ()

  Apply changes associated with character data.

- void revertChdata ()

  Revert changes associated with character data.

- void setChdataAttributes (SXE.Set set)

  Set character data attributes of a set instruction.

Package Functions

- void setChdata (String chdata)

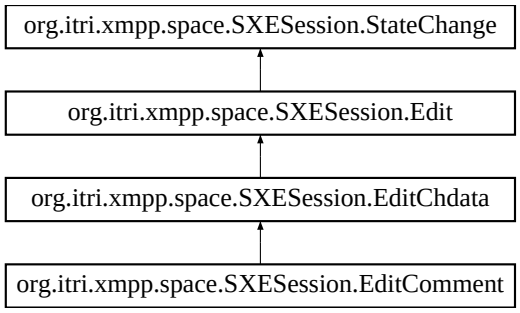
  Set the new character data.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SXESession.java

1.26 org.itri.xmpp.space.SXESession.EditComment Class Reference

State change for changing an comment node. Inheritance diagram for org.itri.xmpp.space.SXESession.EditComment::



Public Member Functions

- SXE.StateChange getStateChange ()

  Get the corresponding state change instruction.

Package Functions

- EditComment (Comment comment, int version)

  Create a new state change for changing a comment.

- void apply () throws OrphanedContent

  Apply the changes to the comment.

- void revert () throws OrphanedContent

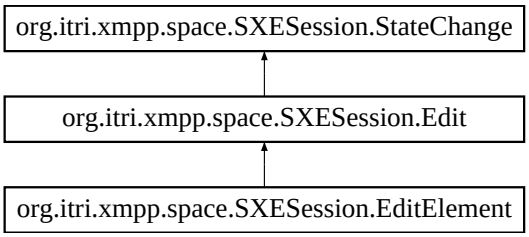
  Revert the changes of the comment back to it's previous state.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SXESession.java

1.27 org.itri.xmpp.space.SXESession.EditElement Class Reference

State change changin an element. Inheritance diagram for org.itri.xmpp.space.SXESession.EditElement::



Public Member Functions

- SXE.StateChange getStateChange ()

  Get the corresponding state change instruction.

Package Functions

- EditElement (Element element, int version)

  Create a new state change for changing an elementö.

- void setName (String name)

  Change the name of the element.

- void setNs (String ns)

Change the namespace of the element.

- void apply () throws OrphanedContent

  Apply the changes to the element.

- void revert () throws OrphanedContent

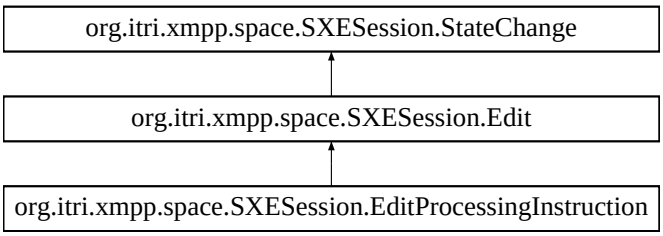
  Revert changes back to to the element initial state.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SXESession.java

1.28 org.itri.xmpp.space.SXESession.EditProcessingInstruction Class Reference

State change for changing a processing instruction node. Inheritance diagram for org.itri.xmpp.space.SXESession.EditProcessingInstruction::



Public Member Functions

- SXE.StateChange getStateChange ()

  Get the corresponding state change instruction.

Package Functions

- EditProcessingInstruction (ProcessingInstruction processingInstruction, int version)

  Create a new state change for changing processing instruction node.

- void setPitarget (String pitarget)

  Change the processing instruction target.

- void setPidata (String pidata)

  Change the processing instruction data.

- void apply () throws OrphanedContent

  Apply the changes to this processing instruction node.

- void revert () throws OrphanedContent

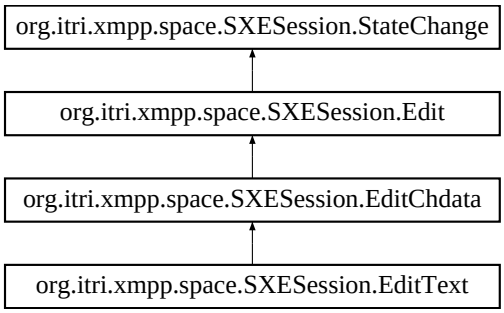
  Revert the changes of this processing instruction node to it's previous state.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SXESession.java

1.29 org.itri.xmpp.space.SXESession.EditText Class Reference

State change for changing a text node. Inheritance diagram for org.itri.xmpp.space.SXESession.EditText::



Public Member Functions

- SXE.StateChange getStateChange ()

  Get the corresponding state change instruction.

Package Functions

- EditText (Text text, int version)

  Create a new state change for changing text node.

- void apply () throws OrphanedContent

  Apply the changes to the text node.

- void revert () throws OrphanedContent

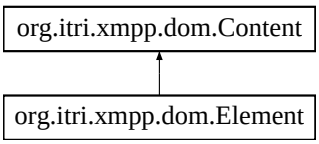
  Revert the changes of the text node to it's previous state.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SXESession.java

1.30 org.itri.xmpp.dom.Element Class Reference

XML Element node. Inheritance diagram for org.itri.xmpp.dom.Element::



Public Member Functions

- Element (String rid, String name, String ns)
  Create a new XML element node.
- Element (String rid, String name)
  Create a new XML element node.
- void setName (String newName)
  Set the name of the element.
- String getName ()
  Get the name of the element.
- void setNs (String newNs)
  Set the namespace of this element.
- String getNs ()
  Get the namespace of this element.
- void addAttribute (Attr attr)
  Add an attribute to this element.
- Attr getAttribute (String prefix, String name)
  Get the attribute given it's name and prefix.
- String getAttributeValue (String prefix, String name)
  Get the attribute value given it's name and prefix.
- Attr getAttribute (String name)
  Get the attribute given it's name.
- boolean hasAttributeWithRid (String rid)

Return true if there is an attribute in this element with the given rid.

- boolean hasAttribute (Attr attr)
  Return true if the given attribute exists in this element.
- String getAttributeValue (String name)
  Get the attribute value given it's name.
- void removeAttribute (Attr attr)
  Remove an attribute from this element.
- Vector< Attr > getAttributeVector ()
  Get all the attributes stored in a vector.
- Attr popFirstAttribute ()
  Pop and return the first attribute of this element.
- void addContent (Content content)
  Add content to this element.
- void removeContent (Content content)
  Remove content from this element.
- boolean hasContentWithRid (String rid)
  Return true if there is an attribute in this element with the given rid.
- Content popFirstContent ()
  Pop and return the first content of this element.
- Vector< Content > getContentVector ()
  Get all the content stored in a vector.
- Content getContentWithType (Class clazz)
  Get the first sub content of the given type.
- String getContentString ()
  Get the first Text sub content.
- String toXML ()
  Output this element and it's sub element as XML code.

Protected Member Functions

- Content _cloneContent ()
  Clone element.

Package Functions

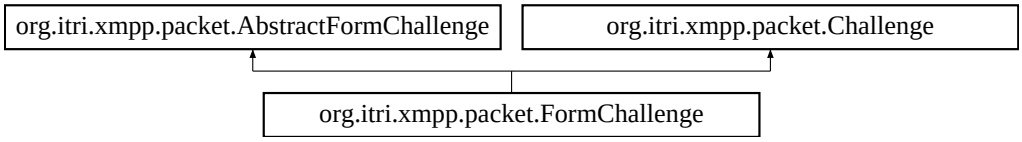
- private< T extends Content > void insertContent (T c, Vector< T > v)
  Insert content into a vector sorted by it's primary weight.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/dom/Element.java

1.31 org.itri.xmpp.packet.FormChallenge Class Reference

A Form challenge. Inheritance diagram for org.itri.xmpp.packet.FormChallenge::



Public Member Functions

- **FormChallenge** (Form form)
- String getElementName ()
  Return the element name which is "challenge".

Static Public Attributes

- static String **ELEMENT_NAME** = "challenge"

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/packet/FormChallenge.java

1.32 org.itri.xmpp.provider.FormChallengeProvider Class Reference

Provider for form challenges and form challenge responses.

Inherits org::jivesoftware::smack::provider::PacketExtensionProvider.

Public Member Functions

- PacketExtension parseExtension (XmlPullParser parser) throws Exception
  Parse the element and sub elements.

Static Package Functions

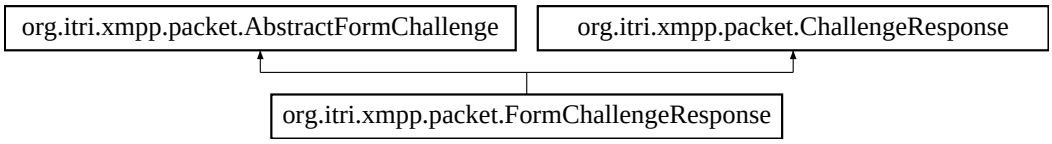
- **[static initializer]**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/provider/FormChallengeProvider.java

1.33 org.itri.xmpp.packet.FormChallengeResponse Class Reference

A Form challenge response. Inheritance diagram for org.itri.xmpp.packet.FormChallengeResponse::



Public Member Functions

- **FormChallengeResponse** (Form form)
- String getElementName ()

  Return the element name which is "response".

Static Public Attributes

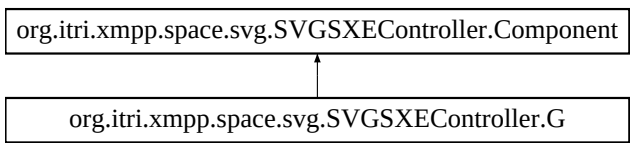
- static final String **ELEMENT_NAME** = "response"

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/packet/FormChallengeResponse.java

1.34 org.itri.xmpp.space.svg.SVGSXEController.G Class Reference

SVG Group <g/>. Inheritance diagram for org.itri.xmpp.space.svg.SVGSXEController.G::



Public Member Functions

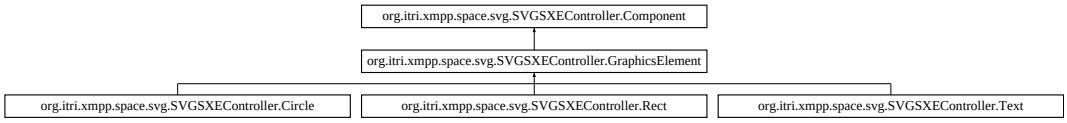
- **G** (Element e)
- Element **generateElement** ()
- void **addComponent** (Component c)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/svg/SVGSXEController.java

1.35 org.itri.xmpp.space.svg.SVGSXEController.GraphicsElement Class Reference

Abstract SVG Graphics element (rect, circle, . Inheritance diagram for org.itri.xmpp.space.svg.SVGSXEController.GraphicsElement::



Public Member Functions

- GraphicsElement (Element e)

  Create a new graphics element.

- **GraphicsElement** (SVGPaint p)
- SVGPaint **getPaint** ()
- float **getStrokeWidth** ()
- Color **getFillColor** ()
- Color **getStrokeColor** ()
- abstract Element **generateShapeElement** ()
- Element **generateElement** ()
- abstract void **move** (float x, float y) throws ClientException

Protected Attributes

- SVGPaint **paint**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/svg/SVGSXEController.java

1.36 org.itri.xmpp.Initializer Class Reference

Initialize the library by loading neccessary classes.

Static Package Functions

- **[static initializer]**
- static void initialize ()

  Initialize the library by loading neccessary classes.

Static Package Attributes

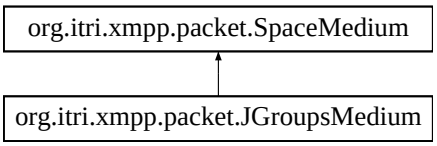
- static Class[ ] **classes**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/Initializer.java

1.37 org.itri.xmpp.packet.JGroupsMedium Class Reference

A medium packet extension for JGroups Spaces. Inheritance diagram for org.itri.xmpp.packet.JGroupsMedium::



Public Member Functions

- **JGroupsMedium** (String cluster)
- String getElementName ()

  Get the element name.

- String getNamespace ()

  Get the namespace which is 'urn:itri:xmpp:space:medium:jgroups'.

- String getCluster ()

  Get the cluster name representing this session.

- String toXML ()

  Convert to an XML String.

Static Public Attributes

- static final String **XMLNS** = "urn:itri:xmpp:space:medium:jgroups"

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/packet/JGroupsMedium.java

1.38 org.itri.xmpp.provider.JGroupsMediumProvider Class Reference

Provider for parsing JGroups medium packet extensions.

Inherits org::jivesoftware::smack::provider::PacketExtensionProvider.

Public Member Functions

- PacketExtension parseExtension (XmlPullParser parser) throws Exception

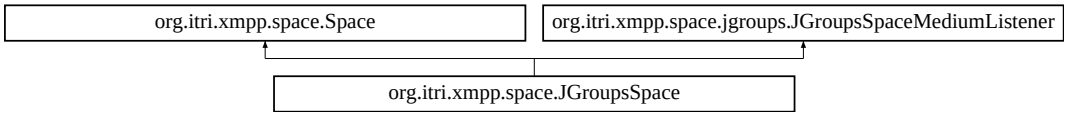
  Parse the element and return a packet extension.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/provider/JGroupsMediumProvider.java

1.39 org.itri.xmpp.space.JGroupsSpace Class Reference

Implementation of Space class for JGroups. Inheritance diagram for org.itri.xmpp.space.JGroupsSpace::



Classes

- class **ActivateThread**
- class **JGroupsPacketReader**

  A thread reading from the group input stream, parses the XML into Smack packets, and calles the process function.

- class **JGroupsSpaceFactory**

  JGroupsSpace factory.

- class **StateMergeThread**

Public Member Functions

- JGroupsSpace (Session session, SXEControllerParams params) throws ClientException

  Create a new JGroups space session, with a new unique id, with a new unused JGroups channel.

- String getNegotiator ()

  Get the negotiator of this space.

- String getMedium ()

  Get medium short name.

- String getAuthentication ()

  Get authentication method.

- int getMembers ()

  Get the amount of participators of this space.

- SpaceMedium getSpaceMedium ()

  Return a SpaceMedium representing this space.

- void **netsplitMerge** ()
- void **closed** ()
- synchronized void crashed (ClientException ce)

  Called when JGroups medium crashed, packet reader crashed, or activation thread crashed.

- void **sendSXE** (SXE sxe) throws ClientException
- void sendPacket (Packet packet) throws ClientException

  Send a packet to the group.

- void handleJoin (SpacePacket sp) throws ClientException

  Handle a <space/> join request sent to this space.

- String synchronousCallback (String message)

  A synchronous callback.

- void **handleGroupMessage** (String message)
- void **spam** ()

Static Public Attributes

- static final String **XMLNS** = "urn:itri:xmpp:space:medium:jgroups"
- static final String **MEDIUM** = "jgroups"
- static final String **DEFAULT_VIRTUAL_MUC** = "room@virtualmuc"

Protected Member Functions

- JGroupsSpace (Session session, String uuid, String clusterName)

  Create a new JGroups space session that connects to an existing one using an existing JGroups channel.

- void _activate () throws ClientException

  Initialize the session.

- void _deactivate ()

  Deactivate the space session.

Static Package Functions

- **[static initializer]**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/JGroupsSpace.java

1.40 org.itri.xmpp.space.jgroups.JGroupsSpaceMedium Class Reference

Inherits org::jgroups::ReceiverAdapter.

Public Member Functions

- **JGroupsSpaceMedium** (String clusterName, String localAddress) throws ClientException
- void **init** () throws ClientException
- void **close** () throws ClientException
- void **setListener** (JGroupsSpaceMediumListener listener)
- void **receive** (Message msg)
- int **getNumberOfParticipants** ()
- void **send** (String content) throws ClientException
- void **viewAccepted** (View newView)
- String **getLocalIdentifier** ()
- boolean **mergeStateRequest** (String content) throws ClientException
- boolean **sendToUnused** (String content) throws ClientException
- void **spam** ()

Public Attributes

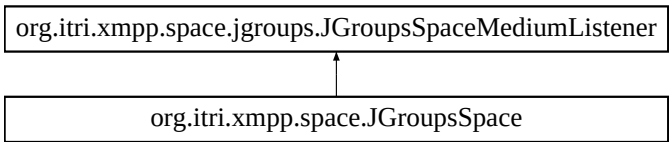
- final String **DEFAULT_MULTICAST_ADDRESS** = "224.8.8.42"
- final int **DEFAULT_PORT** = 50742

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/jgroups/JGroupsSpaceMedium.java

1.41 org.itri.xmpp.space.jgroups.JGroupsSpaceMediumListener Interface Reference

Inheritance diagram for org.itri.xmpp.space.jgroups.JGroupsSpaceMediumListener::



Public Member Functions

- void **closed** ()
- void **crashed** (ClientException ce)
- void **handleGroupMessage** (String message)
- String **synchronousCallback** (String message)
- void **netsplitMerge** ()

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/space/jgroups/JGroupsSpaceMediumListener.java

1.42 org.itri.xmpp.util.JobPool Class Reference

Provides an event based job pool data structure for executing jobs as individual threads and notifying about the progress.

Classes

- class **JobThread**

  A thread containing the job.

- class **TestJob**
- class **TestListener**

Public Member Functions

- void addListener (JobPoolListener listener)
  Add a job progress listener.
- void removeListener (JobPoolListener listener)
  Remove a job progress listener.
- boolean isTerminated ()
  Returns true if the job pool is terminated.
- synchronized void setSelfTerminate (boolean selfTerminate)
  Specify if the job pool should terminate when there is no more jobs running.
- synchronized void add (Callable callable) throws JobPoolException
  Add a job to the pool and report about the jobs progress.

Static Public Member Functions

- static void **main** (String[ ] argv)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/util/JobPool.java

1.43 org.itri.xmpp.util.JobPoolException Class Reference

Job pool exception.

Public Member Functions

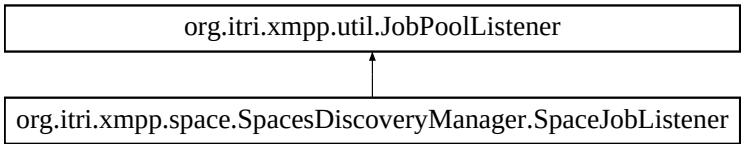
- **JobPoolException** (String msg)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/util/JobPoolException.java

1.44 org.itri.xmpp.util.JobPoolListener Interface Reference

Notifications about progresses of jobs and a job pool. Inheritance diagram for org.itri.xmpp.util.JobPoolListener::



Public Member Functions

- void jobFinished (Callable c, Object result)
  Called when a job finishes returning a value.
- void coughtException (Callable c, Exception e)
  Called when a job crashes because of an exception.
- void poolTerminated (Exception e)
  The job pool terminated.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/util/JobPoolListener.java

1.45 org.itri.xmpp.space.JoinSpace Class Reference

Class for managing the joining procedure of spaces.

Classes

- class **JoinSpaceThread**
  Thread handeling the join procedure.

Public Member Functions

- void setJoinSpaceListener (JoinSpaceListener listener)
  Set join space listener.
- void challengeResponse (ChallengeResponse response)
  Called to respond to a challenge.

Package Functions

- **JoinSpace** (String uuid, SpacePacket.Role role, String entity, JoinSpaceListener listener, Session session)
- void init ()

  Initialize the join procedure.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/JoinSpace.java

1.46 org.itri.xmpp.space.JoinSpaceListener Interface Reference

Notification of space join progress.

Public Member Functions

- void joinFailed (JoinSpace js, String reason)

  Called when joining of a space failed.

- void joinFailed (JoinSpace js, String reason, Exception e)

  Called when joining of a space failed with an exception.

- void joinFinished (JoinSpace js, Space space)

  Called when joining of a space finished.

- void challenge (JoinSpace js, Challenge challenge)

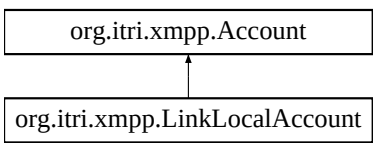
  Called when a challenge is needed to be responded.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/space/JoinSpaceListener.java

1.47 org.itri.xmpp.LinkLocalAccount Class Reference

Link-local account class for XEP-0174 style XMPP sessions. Inheritance diagram for org.itri.xmpp.LinkLocalAccount::



Public Member Functions

- **LinkLocalAccount** (String serviceName)
- String getServiceName ()

  Get the mDNS/DNS-SD service name of this account.

- String getType ()

  Get a human readable name of the account type.

- String getFirstName ()

  Get the first name.

- String getLastName ()

  Get the last name.

- String getEMail ()

  Get the E-mail.

- String getJID ()

  Get the Jabber User ID.

- String getNick ()

  Get the nick name.

- void setFirstName (String firstName)

  Set the first name.

- void setLastName (String lastName)

  Set the last name.

- void setEMail (String email)

  Set the E-mail.

- void setJID (String jid)

  Set the Jabber User ID.

- void setNick (String nick)

  Set the nick name.

- void setPortRange (int min, int max)

  Set the port range the account uses for incoming XEP-0174 connections.

- String **toString** ()

Static Package Functions

- static LinkLocalAccount loadFromXml (XmlPullParser parser, String name) throws XmlPullParserException, IOException

  Create a new account instance by reading an XML block.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/LinkLocalAccount.java

1.48 org.itri.xmpp.LinkLocalChat Class Reference

A class managing a link-local chat.

Inherits org::jivesoftware::smack::LLMessageListener.

Public Member Functions

- LinkLocalChat (LinkLocalSession session, LLChat chat)

  Create a new Link-local chat.

- void sendMessage (String text) throws ClientException

  Send a message to the remote participant.

- String getParticipant ()

  Get the remote participants service name.

- void processMessage (LLChat c, Message m)

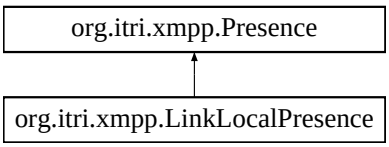
  Process a message from a smack LLChat.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/LinkLocalChat.java

1.49 org.itri.xmpp.LinkLocalPresence Class Reference

Link-local/XEP-0174 presence. Inheritance diagram for org.itri.xmpp.LinkLocalPresence::



Public Member Functions

- String getFirstName ()
  Get the first name associated with this presence.
- String getLastName ()
  Get the last name associated with this presence.
- String getEMail ()
  Get the email associated with this presence.
- String getNick ()
  Get the nick associated with this presence.
- String getJID ()
  Get the JID associated with this presence.
- String **toString** ()

Package Functions

- LinkLocalPresence (LLPresence llPresence)
  Create a new Link-local/XEP-0174 presence.
- LinkLocalPresence (String entity, Type type)
  Create a new Link-local/XEP-0174 presence.
- LinkLocalPresence (String entity)
  Create a new Link-local/XEP-0174 presence with a default type.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/LinkLocalPresence.java

1.50 org.itri.xmpp.LinkLocalSession Class Reference

The implementation of Link-local sessions (using XEP-0174 from Smack API).

Inherits org::itri::xmpp::Session< LinkLocalAccount, LinkLocalPresence >.

Classes

- class **MyLLChatListener**
  Private LLChatListener.

Public Member Functions

- **LinkLocalSession** (LinkLocalAccount account) throws ClientException
- **LinkLocalSession** (LinkLocalAccount account, InetAddress addr) throws ClientException
- **LinkLocalSession** (LinkLocalAccount account, LinkLocalPresence linkLocalPresence) throws ClientException
- **LinkLocalSession** (LinkLocalAccount account, LinkLocalPresence linkLocalPresence, InetAddress addr) throws ClientException
- void addLinkLocalStateListener (LinkLocalStateListener listener)

  Add a new link-local state listener.

- void removeLinkLocalStateListener (LinkLocalStateListener listener)

  Remove link-local state listener.

- void **unknownOriginMessage** (Message m)
- void serviceClosedOnError (Exception e)

  Smack-interface function.

- void serviceClosed ()

  Smack-interface function.

- void serviceNameChanged (String newName, String oldName)

  Smack-interface function.

- void addPacketListener (PacketListener listener, PacketFilter filter)

  Add a packet listener.

- void removePacketListener (PacketListener listener)

  Remove a packet listener.

- void updatePresence (Presence presence) throws ClientException

  Presence updating.

- void updatePresence (Presence.Mode mode) throws ClientException

  Presence updating.

- void updatePresence (String message) throws ClientException

  Set new presence message.

- void updatePresence (Presence.Mode mode, String message) throws ClientException

  Set new presence mode and message.

- void sendPacket (Packet packet) throws ClientException

  Send a packet.

- IQ getIQResponse (IQ request) throws ClientException
  Perform a IQ request and get the response.
- DiscoverInfo discoverInfo (String entity) throws ClientException
  Discover info on the given entity.
- void addFeature (String feature)
  Add Service Discovery feature to this session.
- void **spam** ()
- String **toString** ()

Protected Member Functions

- Chat newChat (String entity) throws ClientException
  Create a new chat given a service name.

Package Functions

- void start () throws ClientException
  Start the Link-local session.
- void stop () throws ClientException
  Stop the Link-local session (WARNING! Not implemented yet.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/LinkLocalSession.java

1.51 org.itri.xmpp.LinkLocalStateListener Interface Reference

Notifications about the state of a link local session.

Public Member Functions

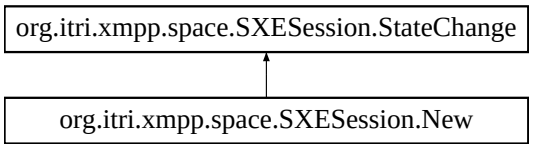
- void serviceNameChanged (String newName, String oldName)
  Called when the mDNS/DNS-SD service name is changed.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/LinkLocalStateListener.java

1.52 org.itri.xmpp.space.SXESession.New Class Reference

State change that creates a new node. Inheritance diagram for org.itri.xmpp.space.SXESession.New::



Public Member Functions

- SXE.StateChange getStateChange ()

  Get the state change instruction for this state change.

- String getTarget ()

  Get the target ID.

Package Functions

- New (SXE.New n)

  Create a new state change that creates a new node.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SXESession.java

1.53 org.itri.xmpp.NewChatListener Interface Reference

Interface for listening for new chat sessions.

Public Member Functions

- void newChat (Chat chat)

  A new chat has been created.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/NewChatListener.java

1.54 org.itri.xmpp.space.NewSpaceListener Interface Reference

Public Member Functions

- void newSpace (Space space)

  Called when a new space has been created, but before it has been initialized.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/space/NewSpaceListener.java

1.55 org.itri.xmpp.space.PasswordForm Class Reference

Functions for using the predescripted password form.

Static Public Member Functions

- static Form passwordForm ()

  Create a new password form.

- static boolean isPasswordForm (Form form)

  Return true if the provided form is a password form.

- static void **setPasswordInPasswordForm** (String password, Form form)
- static String getPassword (Form form)

  Return the password from the form.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/PasswordForm.java

1.56 org.itri.xmpp.dom.Path Class Reference

Class used for creating and evaluating XML Paths.

Classes

- class **ElementNameTurn**

  A turn matching elements given a name.

- class **IsTextTurn**

  A turn matching text contents.

- class **SubAttributeTurn**

  Abstract class for a sub-attribute turn.

- class **SubContentTurn**

  Abstract class for a sub-content turn.

- class **Turn**

  Abstract class for a turn.

Public Member Functions

- Path (Element root)

  Create a new path beginning on a specified root element.

- Content evaluate ()

  Evaluate the path expression and return the resulting content.

- Path element (String name)

  Match next content as an element with a given name.

- Path text ()

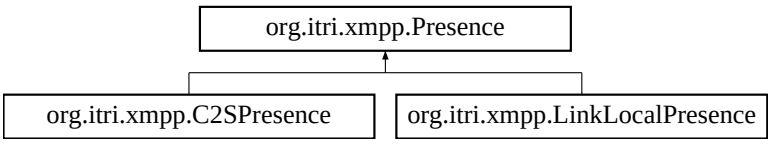
  Match next content as a text content.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/dom/Path.java

1.57 org.itri.xmpp.Presence Class Reference

Abstract class for providing basic presence functionality. Inheritance diagram for org.itri.xmpp.Presence::



Public Types

- enum **Mode** {
  chat, **available**, **away**, **xa**,
  dnd }
- enum **Type** { **available**, **unavailable** }

Public Member Functions

- Type getType ()
  Return what type of presence.
- void setSession (Session session)
  Set what session this presence is associated with.
- Session getSession ()
  Return the session that this presence comes from.
- boolean isUnavailablePresence ()
  Returns true if this presence is an 'unavailable' presence.
- String getMessage ()
  Get the message bundled with this presence.
- Mode getMode ()
  Get the mode bundled with this presence.
- String getEntityName ()
  Get the entity name of this presence.

Protected Member Functions

- **Presence** (String entity, Mode mode, String message)
- **Presence** (String entity, Type type, String message)
- **Presence** (String entity, Type type)
- **Presence** (String entity, String message)
- **Presence** (String entity, Mode mode)
- **Presence** (String entity)

Protected Attributes

- Contact **contact**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/Presence.java

1.58 org.itri.xmpp.PresenceListener< PresenceType extends Presence > Interface Reference

Notificate about presences of contacts being changed.

Public Member Functions

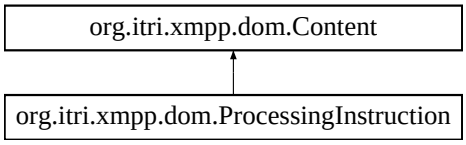
- void presenceUpdated (PresenceType presence)
  A new presence event has been received.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/PresenceListener.java

1.59 org.itri.xmpp.dom.ProcessingInstruction Class Reference

Processing instruction XML node. Inheritance diagram for org.itri.xmpp.dom.ProcessingInstruction::



Public Member Functions

- ProcessingInstruction (String rid, String target, String data)
  Create a new processing instruction XML node.
- void setPitarget (String pitarget)
  Set the processing instruction target.
- String getPitarget ()
  Get the processing instruction target.
- void setPidata (String pidata)
  Set the processing instruction data.
- String getPidata ()
  Get the processing instruction data.
- String toXML ()
  Generate XML code for this XML node.

Public Attributes

- String **target**
- String **data**

Protected Member Functions

- Content _cloneContent ()

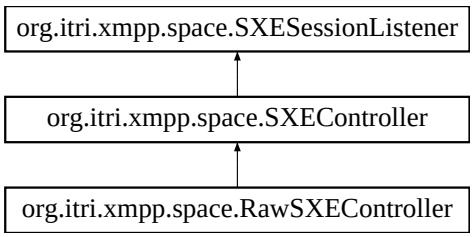
  Clone this content.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/dom/ProcessingInstruction.java

1.60 org.itri.xmpp.space.RawSXEController Class Reference

Inheritance diagram for org.itri.xmpp.space.RawSXEController::



Classes

- class **RawSXEControllerFactory**

Public Member Functions

- void **parseDOM** () throws ClientException

Protected Member Functions

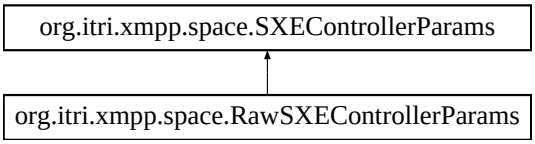
- **RawSXEController** (SXESession sxeSession, Space space)
- String **getDocumentType** ()
- String **getDocumentHead** ()
- Element **rootElement** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/RawSXEController.java

1.61 org.itri.xmpp.space.RawSXEControllerParams Class Reference

Inheritance diagram for org.itri.xmpp.space.RawSXEControllerParams::



Public Member Functions

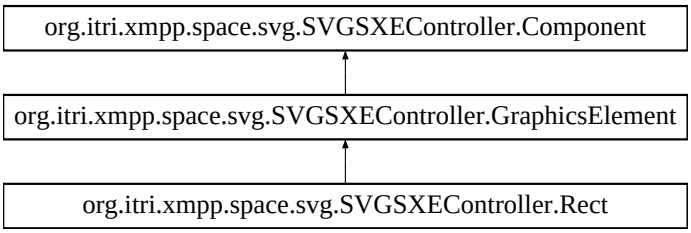
- SXEControllerFactory **getSXEControllerFactory** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/RawSXEControllerParams.java

1.62 org.itri.xmpp.space.svg.SVGSXEController.Rect Class Reference

SVG Rect. Inheritance diagram for org.itri.xmpp.space.svg.SVGSXEController.Rect::



Public Member Functions

- **Rect** (Element e) throws ClientException
- **Rect** (float x, float y, float width, float height, SVGPaint paint)
- float **getX** ()
- float **getY** ()
- float **getWidth** ()
- float **getHeight** ()
- void move (float x, float y) throws ClientException

  Move the rectangle.

- Element **generateShapeElement** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/svg/SVGSXEController.java

1.63 org.itri.xmpp.Session< AccountType extends Account, PresenceType extends Presence > Class Reference

A session is general functions which may be used on all kinds of XMPP sessions, both c2s and link-local.

Public Types

- enum **SessionState** { **inactive**, **activating**, **active** }

Public Member Functions

- boolean isActivating ()

  Return true if the session is currently activating.

- boolean isInactive ()

  Return true if the session is currently inactive.

- boolean isActive ()

  Return true if the session is currently active.

- SessionState getCurrentState ()

  Return the current session state.

- String getAccountName ()

  Get the account name associated with this session.

- PresenceType getOwnPresence ()

  Get the presence for this session.

- PresenceType getPresence (String entity)

  Get the presence associated with a entity name.

- Collection< PresenceType > getPresences ()

  Return a collection of all known presences.

- synchronized void notifyAboutAllPresences (PresenceListener listener)

  Notify presence listener about all known presences.

- void addPresenceListener (PresenceListener presenceListener)

  Add a presence listener.

- synchronized void **addPresenceListenerAndGetAll** (PresenceListener presenceListener)
- void removePresenceListener (PresenceListener presenceListener)

  Remove presence listener.

- void addSessionStateListener (SessionStateListener listener)

  Add a session state listener.

- void removeSessionStateListener (SessionStateListener listener)

  Remove a session state listener.

- void notifySessionClosed ()

  Notification that the session was closed.

- void notifySessionCrashed (ClientException ce)

  Notification that the session crashed.

- void addNewChatListener (NewChatListener listener)

  Add a new-chat listener.

- void removeNewChatListener (NewChatListener listener)

  Remove a new-chat listener.

- Collection< Chat > getChats ()

  Get all chats of this session.

- Chat getChat (String entity) throws ClientException

  Get a chat given a entity (serivce name or JID).

- abstract void updatePresence (Presence presence) throws ClientException

  Update presence.

- abstract void updatePresence (Presence.Mode mode) throws ClientException

  Update presence.

- abstract void updatePresence (String message) throws ClientException

  Update presence.

- abstract void updatePresence (Presence.Mode mode, String message) throws ClientException

  Update presence.

- abstract void addPacketListener (PacketListener listener, PacketFilter filter)

  Add a packet listener with a given filter.

- abstract void removePacketListener (PacketListener listener)

  Remove a packet listener.

- abstract void sendPacket (Packet packet) throws ClientException

  Send a packet.

- abstract IQ getIQResponse (IQ iq) throws ClientException

  Perform a IQ request and get the response.

- abstract DiscoverInfo discoverInfo (String entity) throws ClientException

  Discover info on the given entity.

- abstract void addFeature (String feature)

  Add Service Discovery feature to this session.

- abstract void **spam** ()

Protected Member Functions

- **Session** (AccountType account, PresenceType presence)
- void setActivating ()

  Set the session state to activating.

- void setInactive ()

  Set the session state to inactive.

- void setActive ()

  Set the session state to active.

- synchronized void presenceUpdated (PresenceType presence)

  Called when a presence update has been noticed.

- void addChat (String participant, Chat chat)

  Add chat session.

- abstract Chat newChat (String entity) throws ClientException

  Create a new chat given an entity (service name/JID).

Protected Attributes

- AccountType **account**
- PresenceType **presence**
- SessionState **sessionState** = SessionState.inactive
- Map< String, PresenceType > presences

  Map of presences (full jid -> presence info).

Package Functions

- abstract void start () throws ClientException
  Start the session.
- abstract void stop () throws ClientException
  Stop the session.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/Session.java

1.64 org.itri.xmpp.SessionListener< SessionType extends Session > Interface Reference

Notificate about session states.

Public Member Functions

- void sessionActivating (SessionType session)
  Called when a session is activating.
- void sessionActivated (SessionType session)
  Called when a session has finished activating.
- void sessionDeactivated (SessionType session)
  Called when a session has been deactivated.
- void sessionActivationFailed (SessionType session, ClientException ce)
  Called when a session activation failed due to an exception.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/SessionListener.java

1.65 org.itri.xmpp.SessionManager Class Reference

Manages session initiation of sessions.

Classes

- class **SessionStarter**
  A thread used for activating sessions in the background.

Public Member Functions

- synchronized void startSession (Session session) throws ClientException
  Start a new session and notify about its progress.
- synchronized void stopSession (String name) throws ClientException
  Stop a session.
- synchronized void stopSession (Session session) throws ClientException
  Stop a session.
- synchronized void stopAllSessions () throws ClientException
  Stop all sessions.
- synchronized boolean isAuthorized (Session session)
  Return true if the given session is allowed to be started.
- synchronized Collection< Session > getSessions ()
  Get a collection of all sessions.
- synchronized int numberOfSessions ()
  Get the number of sessions.
- synchronized int numberOfSessionsOfType (Class clazz)
  Get the number of sessions of a specified type.
- synchronized int numberOfActiveSessions ()
  Get the number of active sessions.
- synchronized int numberOfActiveSessionsOfType (Class clazz)
  Get the number of active sessions of a specified type.
- synchronized void addSessionListener (SessionListener listener)
  Add a session listener.
- synchronized void removeSessionListener (SessionListener listener)
  Remove a session listener.
- synchronized void addContactListener (ContactListener listener)
  Add global contact listener.
- synchronized void addContactListenerAndGetAll (ContactListener listener)
  Add a contact listener and get all already known contacts.
- synchronized void addPresenceListenerAndGetAll (PresenceListener listener)
  Add a presence listener and get all already known presences.

- synchronized void addPresenceListener (PresenceListener listener)

  Add global presence listener.

- synchronized void removeContactListener (ContactListener listener)

  Remove a global contact listener.

- synchronized void removePresenceListener (PresenceListener listener)

  Remove a global presence listener.

- synchronized void updatePresence (Presence presence) throws ClientException

  Update presence.

- synchronized void updatePresence (Presence.Mode mode) throws ClientException

  Update presence.

- synchronized void updatePresence (String message) throws ClientException

  Update presence.

- synchronized void updatePresence (Presence.Mode mode, String message) throws ClientException

  Update presence.

- synchronized LinkLocalSession getLinkLocalSession ()

  Get the LinkLocalSession associated with this manager, if any.

- synchronized ChatManager getChatManager ()

  Get the chat manager of this session manager.

Static Public Member Functions

- static synchronized SessionManager getInstance ()

  Get the SessionManager instance.

Static Package Functions

- **[static initializer]**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/SessionManager.java

1.66 org.itri.xmpp.SessionStateListener Interface Reference

Notifications about the state of a session.

Public Member Functions

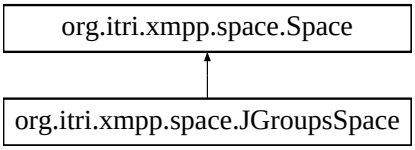
- void sessionActivated ()
  The session was activated.
- void sessionClosed ()
  The session was closed.
- void sessionCrashed (ClientException ce)
  The session crashed due to an exception.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/SessionStateListener.java

1.67 org.itri.xmpp.space.Space Class Reference

An abstract class reperesenting basic and required functionality for a space. Inheritance diagram for org.itri.xmpp.space.Space::



Public Member Functions

- Spaces getSpaces ()
  Get the spaces manager of this space.
- SXEController getSXEController ()
  Get the SXEController of this space.
- void addSpaceListener (SpaceListener listener)
  Add space listener.
- void removeSpaceListener (SpaceListener listener)
  Remove space listener.

- void **notifySpaceContentInvalid** (ClientException ce)
- SXESession getSXESession ()

  Get the SXESession instance of this space session.

- String getUUID ()

  Return the Universal Unique IDentifier for this space.

- String getName ()

  Return the name of this space.

- String getInfo ()

  Return the info of this space.

- void setName (String name)

  Set the name of this space.

- void setInfo (String info)

  Set the info of this space.

- String getContentType ()

  Get the content type of this space.

- void **setPasswordProtection** (String password)
- boolean isPasswordProtected ()

  Returns true if this space is protected with trivial password protection.

- abstract String getNegotiator ()

  Get the negotiator of this space.

- abstract String getMedium ()

  Get medium short name.

- abstract String getAuthentication ()

  Get authentication method.

- abstract int getMembers ()

  Get the amount of participators of this space.

- abstract SpaceMedium getSpaceMedium ()

  Return a SpaceMedium representing this space.

- void activate () throws ClientException

  Activate the session.

- void deactivate ()

  Deactivate the space session.

- boolean isActive ()
  Returns true if this space session is activated.
- boolean isActivating ()
  Returns true if this space is currently activating.
- boolean isInactive ()
  Returns true if this space is currently inactivate.
- abstract void handleJoin (SpacePacket sp) throws ClientException
  Handle a <space/> join request sent to this space.
- void handleBasicJoin (SpacePacket sp) throws ClientException
  Handle a basic join.
- void handleCasualJoin (SpacePacket sp) throws ClientException
  Handle casual join.
- void handlePasswordJoin (SpacePacket sp) throws ClientException
  Handle password join.
- SpaceAds.Description getDescription ()
  Return a SpaceAds.Description element.
- SpaceAds.Ad toElement ()
  Convert this space to an space advertisement element.
- abstract void **sendSXE** (SXE sxe) throws ClientException
- abstract void **spam** ()

Protected Types

- enum **State** { **inactive**, **activating**, **active** }

Protected Member Functions

- **Space** (Session session, SXEControllerParams sxeControllerParams)
- **Space** (Session session, String uuid)
- void **spaceInitiated** ()
- void notifySpaceInitiated ()
  Notify that a space has been initiated and is ready for use.
- void notifySpaceCrashed (ClientException ce)
  Notify that a space has crashed with an exception.

- void notifySpaceClosed ()

  Notify that a space has closed.

- void notifySpaceInitiationFailed (ClientException ce)

  Notify that a space has closed.

- abstract void _activate () throws ClientException

  The protected activation function.

- abstract void _deactivate ()

  Deactivate the space session.

Protected Attributes

- String **uuid** = UUID.randomUUID().toString()
- SXESession **sxeSession**
- SXEController **sxeController**
- Session **session**
- State **state** = State.inactive

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/Space.java

1.68 org.itri.xmpp.packet.SpaceAds Class Reference

Space Advertisement packet.

Inherits org::jivesoftware::smack::packet::IQ.

Classes

- class **Ad**

  An advertisement for a space.

- class **Description**

  A description of a space.

Public Member Functions

- **SpaceAds** (List< SpaceAds.Ad > ads)
- List< SpaceAds.Ad > getAds ()

  Return a list of all the space ads.

- String getChildElementXML ()
  Return the child element XML as a string.

Static Public Attributes

- static final String **ELEMENT_NAME** = "spaces"

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/packet/SpaceAds.java

1.69 org.itri.xmpp.provider.SpaceAdsProvider Class Reference

Provider for parsing <spaces/> packets.

Inherits org::jivesoftware::smack::provider::IQProvider.

Public Member Functions

- IQ parseIQ (XmlPullParser parser) throws Exception
  Parse a <spaces/> packet.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/provider/SpaceAdsProvider.java

1.70 org.itri.xmpp.space.SpaceFactory Class Reference

A class used for managing space factories given <medium/> elements.

Static Public Member Functions

- static void addSpaceFactory (String xmlns, SpaceFactoryInterface factory)
  Add a space factory to this manager.
- static Space createSpaceFromPacket (SpacePacket packet, Session session) throws ClientException
  Given a <space/> packet, create a new space instance that joins joins the space.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SpaceFactory.java

1.71 org.itri.xmpp.space.SpaceFactoryInterface Interface Reference

An interface for space factories.

Inherited by org.itri.xmpp.space.JGroupsSpace.JGroupsSpaceFactory.

Public Member Functions

- Space spawn (SpacePacket packet, Session session) throws ClientException
  Given a <space/> packet use the medium element to create a new space instance.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/space/SpaceFactoryInterface.java

1.72 org.itri.xmpp.space.SpaceInfo Class Reference

Provides information for a collaborative work space.

Public Member Functions

- **SpaceInfo** (String UUID)
- String **getUUID** ()
- boolean hasDescription ()
  Returns true if a description was provided.
- SpaceAds.Description getDescription ()
  Returns the description of this space provided by the advertising entity.
- void setDescription (SpaceAds.Description description)
  Set the description of this space.
- Set< String > getSpaceContacts ()
  Get the set of contacts that advertised this space.
- String **toString** ()

Package Functions

- void addSpaceContact (String entity)
  Add a contact that advertises this space.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SpaceInfo.java

1.73 org.itri.xmpp.space.SpacesDiscoveryManager.SpaceJob Class Reference

A SpaceJob is added to a JobPool and it is used for retrieving space ads from users.

Inherits java::util::concurrent::Callable< SpaceAds >.

Public Member Functions

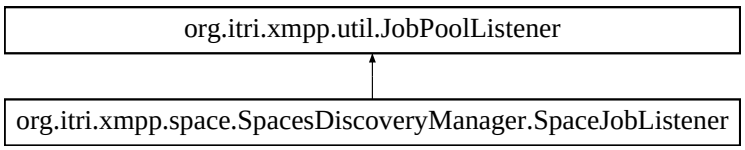
- **SpaceJob** (String user)
- String **getUser** ()
- SpaceAds **call** () throws Exception

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SpacesDiscoveryManager.java

1.74 org.itri.xmpp.space.SpacesDiscoveryManager.SpaceJobListener Class Reference

SpaceJobListener listenes for finished jobs from the pool. Inheritance diagram for org.itri.xmpp.space.SpacesDiscoveryManager.SpaceJobListener::



Public Member Functions

- void jobFinished (Callable job, Object ads)

  Called when a job finishes returning a value.

- void coughtException (Callable sj, Exception e)

  Called when a job crashes because of an exception.

- void poolTerminated (Exception e)

  The job pool terminated.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SpacesDiscoveryManager.java

1.75 org.itri.xmpp.space.SpaceListener Interface Reference

Notifications about spaces.

Inherited by org.itri.xmpp.sample.TestUI.TestSpaceListener.

Public Member Functions

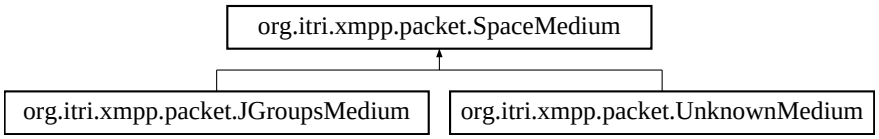
- void spaceInitiated ()
  Called when a space has been initiated and is ready for use.
- void spaceInitiationFailed (ClientException ce)
  Called when a space failed to initiate completely.
- void spaceCrashed (ClientException ce)
  Called when a space has crashed.
- void spaceClosed ()
  Called when a space is closed.
- void spaceContentInvalid (ClientException ce)
  Called when the content of a space is invalid.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/space/SpaceListener.java

1.76 org.itri.xmpp.packet.SpaceMedium Interface Reference

Interface for space mediums. Inheritance diagram for org.itri.xmpp.packet.SpaceMedium::



The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/packet/SpaceMedium.java

1.77 org.itri.xmpp.packet.SpacePacket Class Reference

Packet used for negotiating participation in a space.

Inherits org::jivesoftware::smack::packet::IQ.

Public Types

- enum **State** { **join**, **challenge**, **result** }
- enum **Role** { **participant**, **visitor** }

Public Member Functions

- **SpacePacket** (String UUID, State state, Role role)
- void setChallenge (Challenge challenge)

  Set the challenge.

- Challenge getChallenge ()

  Get the challenge.

- void setResponse (ChallengeResponse response)

  set the response.

- ChallengeResponse getResponse ()

  Get the response.

- void setMedium (SpaceMedium medium)

  Set the medium.

- SpaceMedium getMedium ()

  Get the medium.

- State getState ()

  Get the state this space packet is in.

- Role getRole ()

  Get the role being negotiated.

- String getUUID ()

  Get the UUID of the space being negotiated about.

- String getChildElementXML ()

  Return a XML string containing the space IQ child element.

Static Public Attributes

- static final String **XMLNS** = "urn:itri:xmpp:space"
- static final String **ELEMENT_NAME** = "space"

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/packet/SpacePacket.java

1.78 org.itri.xmpp.provider.SpacePacketProvider Class Reference

Provider for parsing <space/> packets.

Inherits org::jivesoftware::smack::provider::IQProvider.

Public Member Functions

- IQ parseIQ (XmlPullParser parser) throws Exception
  Parse a <space/> packet.

Static Public Member Functions

- static void addProviders ()
  Add providers to Spaces.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/provider/SpacePacketProvider.java

1.79 org.itri.xmpp.space.Spaces Class Reference

Class used for retrieving space advertisements from users and to advertise own spaces to other users.

Classes

- class SpacesListener
  A class processing <spaces/> packets.

Public Member Functions

- void processSpacePacket (SpacePacket sp)
  Process a <space/> IQ request.
- void addNewSpaceListener (NewSpaceListener listener)
  Add listener for new spaces.
- void removeNewSpaceListener (NewSpaceListener listener)
  Remove new space listener.
- void startSpace (Space space) throws ClientException

Add a space and start to initialize it.

- boolean hasSpace (String uuid)

  Returns true if a space with a given UUID exists.

- Space getSpace (String uuid)

  Get the space associated with the given UUID.

- SpaceAds getSpaceAdsFromUser (String entity) throws ClientException

  Download space advertisements from a given entity.

- JoinSpace joinSpace (SpaceInfo info, String entity, JoinSpaceListener listener) throws ClientException

  Join space using a SpaceInfo object discovered when searching.

- JoinSpace joinSpace (String UUID, String entity, JoinSpaceListener listener) throws ClientException

  Join space using a UUID string.

- SpaceAds toSpaceAds ()

  Generate a Spaces packet extension representing the ongoing sessions.

- void **spam** ()

Static Public Member Functions

- static Spaces getInstanceFor (Session session)

  Return the instance for a given session.

- static void addDOMControllerFactory (String documentType, SXEController-Factory factory)

  Add document structure controller.

- static SXEController getSXEController (String contentType, Space space, SXE-Session sxeSession)

  Get the document structure controller for a certain content type.

- static void addChallengeProvider (String xmlns, PacketExtensionProvider provider)

  Add provider for parsing <challenge/> package extensions.

- static void addResponseProvider (String xmlns, PacketExtensionProvider provider)

  Add provider for parsing <response/> package extensions.

- static void addMediumProvider (String xmlns, PacketExtensionProvider provider)

Add provider for parsing <medium/> package extensions.

- static PacketExtensionProvider getChallengeProvider (String xmlns)
  Get the provider that parser that parses challenges of the given namespace.
- static PacketExtensionProvider getResponseProvider (String xmlns)
  Get the provider that parser that parses responses of the given namespace.
- static PacketExtensionProvider getMediumProvider (String xmlns)
  Get the provider that parser that parses mediums of the given namespace.

Static Public Attributes

- static final String **XMLNS** = "urn:itri:xmpp:space"
- static final String **FEATURE** = "urn:itri:xmpp:space"

Static Package Functions

- **[static initializer]**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/Spaces.java

1.80 org.itri.xmpp.space.SpacesDiscoveryManager Class Reference

Handles discovering and searching of spaces.

Classes

- class **SearchInitiator**
  This class is used for initating a search.
- class SpaceJob
  A SpaceJob is added to a JobPool and it is used for retrieving space ads from users.
- class SpaceJobListener
  SpaceJobListener listenes for finished jobs from the pool.

Public Member Functions

- **SpacesDiscoveryManager** (Session session)
- void initiateSearch (SpaceSearchListener listener) throws ClientException

  Initiate a search for available spaces.

- boolean isSearching ()

  Returns true if a search is already being performed.

- SpaceInfo getSpaceInfo (String UUID)

  Get a space info associated with a specific identifier.

- Collection< SpaceInfo > getSpaceInfos ()

  All spaces found during the previous search.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SpacesDiscoveryManager.java

1.81 org.itri.xmpp.space.SpaceSearchListener Interface Reference

Notification about space search process.

Public Member Functions

- void spaceDiscovered (SpaceInfo spaceInfo)

  A new space discovered.

- void spaceContactAdded (SpaceInfo spaceInfo, String entityID)

  A new contact added to a space.

- void coughtException (ClientException e)

  Cought an exception while initiating probes.

- void searchAborted (ClientException e)

  The search is aborted.

- void searchFinished ()

  The search has finished and all result has been reported.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/space/SpaceSearchListener.java

1.82 org.itri.xmpp.space.Spaces.SpacesListener Class Reference

A class processing <spaces/> packets.

Inherits org::jivesoftware::smack::PacketListener.

Public Member Functions

- void processPacket (Packet packet)

  Process a <space/> packet.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/Spaces.java

1.83 org.itri.xmpp.packet.SXE.StateChange Interface Reference

A state change instruction.

Inherited by org.itri.xmpp.packet.SXE.New, org.itri.xmpp.packet.SXE.Remove, and org.itri.xmpp.packet.SXE.Set.

Public Member Functions

- String toXML ()

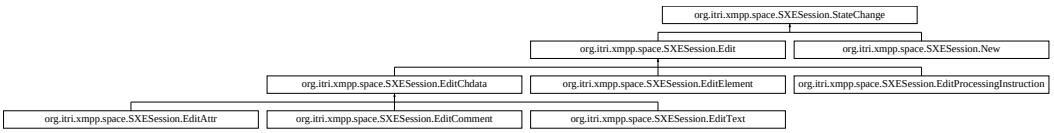
  Generate the XML output of this instruction.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/packet/SXE.java

1.84 org.itri.xmpp.space.SXESession.StateChange Interface Reference

Interface for state changes. Inheritance diagram for org.itri.xmpp.space.SXESession.StateChange::



Public Member Functions

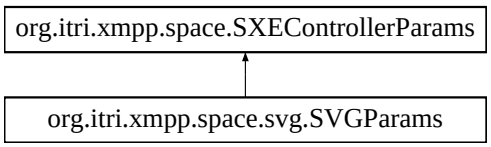
- SXE.StateChange getStateChange ()
  Get the corresponding StateChange packet instruction.
- String getTarget ()
  Get the target ID.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/space/SXESession.java

1.85 org.itri.xmpp.space.svg.SVGParams Class Reference

Class containing parameters for initiating a whiteboard space. Inheritance diagram for org.itri.xmpp.space.svg.SVGParams::



Public Member Functions

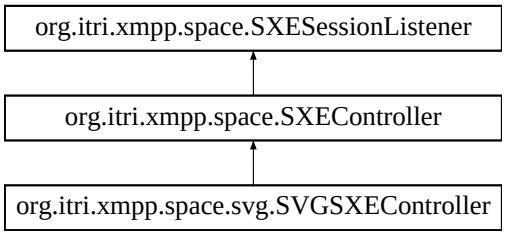
- SVGParams (int width, int height, String name, String info)
  Create a new svg param class.
- int getWidth ()
  Get the width of the view box.
- int getHeight ()
  Get the height of the view box.
- String getName ()
  Get the name of the space.
- String getInfo ()
  Get the description of the space.
- SXEControllerFactory **getSXEControllerFactory** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/svg/SVGParams.java

1.86 org.itri.xmpp.space.svg.SVGSXEController Class Reference

SXE XML DOM controller for editing a Scalable Vector Graphics tree. Inheritance diagram for org.itri.xmpp.space.svg.SVGSXEController::



Classes

- class Circle

  SVG Circle.

- class **Color**

  SVG Color.

- class Component

  Abstract class for a SVG tree component.

- class **FontSize**

  SVG Font size class.

- class G

  SVG Group <g/>.

- class GraphicsElement

  Abstract SVG Graphics element (rect, circle, .

- class Rect

  SVG Rect.

- class **SVGPaint**

  A SVG Paint.

- class **SVGSXEControllerFactory**
- class Text

  SVG Text.

Public Member Functions

- SVGSXEController (SXESession sxeSession, Space space)

  Create a new controller ment for joining an existing session.

- SVGSXEController (SXESession sxeSession, Space space, SVGParams svgParams)

  Create a new controller ment for creating a new session.

- String getMimeType ()

  Get the MIME-Type (image/svg+xml).

- float getMinX ()

  Get the min x of the view box.

- float getMinY ()

  Get the min y of the view box.

- float getWidth ()

  Get the width of the view box.

- float getHeight ()

  Get the height of the view box.

- boolean handlesName ()

  Return true if this sxe session handles session name.

- String getName ()

  Return the name of this SXE session.

- boolean handlesInfo ()

  Return true if this sxe session handles session description.

- String getInfo ()

  Return the description of this sxe session.

- List< GraphicsElement > getGraphicsElements ()

  Get a list of all graphics elements of this SVG Image.

- void newGraphicsElement (GraphicsElement ge) throws ClientException

  Create a new graphic element.

- void resizeGraphicsElement (GraphicsElement ge, float x, float y, float width, float height) throws ClientException

  Resize a graphic element.

- void resizeRect (Rect rect, float x, float y, float width, float height) throws ClientException

  Resize a rectangle.

- void resizeCircle (Circle circle, float r) throws ClientException

  Resize a circle.

- void alterText (Text text, String string, FontSize fontSize) throws ClientException

  Change the text of a text element.

- void changePaint (GraphicsElement ge, SVGPaint paint) throws ClientException

  Change the paint of a graphics element.

- void removeGraphicsElement (GraphicsElement ge) throws ClientException

  Remove a graphics element.

- void parseDOM () throws ClientException

  Parse the internal DOM tree regeneretating the graphics element.

Static Public Attributes

- static final String **XMLNS** = "http://www.w3.org/2000/svg"
- static final String **MIME_TYPE** = "image/svg+xml"

Protected Member Functions

- String getDocumentType ()

  Return the document type (same as getMimeType()).

- String getDocumentHead ()

  Get the document head.

- Element rootElement ()

  Get the root element of an SVG tree.

Static Package Functions

- **[static initializer]**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/svg/SVGSXEController.java

1.87 org.itri.xmpp.packet.SXE Class Reference

Shared XML Editing packet.

Inherits org::jivesoftware::smack::packet::PacketExtension.

Classes

- class **New**
  An instruction creating a new DOM node.
- class **NewAttr**
  Instruction for creating a new attribute.
- class **NewComment**
  Instruction for creating a new comment.
- class **NewElement**
  An instruction creating a new DOM element node.
- class **NewProcessingInstruction**
  Instruction for creating processing instruction nodes.
- class **NewText**
  Instruction for creating new Text nodes.
- class **Remove**
  Instruction for removing a DOM node.
- class **Set**
  Instruction for changing an already existing XML DOM node.
- class **State**
  Representation of the state, except the state change instructions.
- interface StateChange
  A state change instruction.

Public Types

- enum Type {
  element, **text**, **attr**, **comment**,
  processinginstruction }
  Edit instruction types.

Public Member Functions

- SXE (String sessionId, String sxeId)
  Create a new SXE packet.
- void setStateRequest ()
  Set state request flag to true.
- boolean isStateRequest ()
  Return true if this packet is a state request packet.
- String getSessionId ()
  Get hte session id of this packet.
- String getSXEId ()
  Get the SXE id of this set of instructions.
- void addStateChange (StateChange sc)
  Add a state change instruction.
- void setStateChanges (List< StateChange > sts)
  Set the state change instructions.
- List< StateChange > getStateChanges ()
  Get all state change instructions.
- void setState (State state)
  Set the state.
- State getState ()
  Get the state.
- boolean isState ()
  Return true if this packet is a state packet.
- String getElementName ()
  Returns the root element name.
- String getNamespace ()
  Returns the root element XML namespace.
- String toXML ()
  Returns the XML reppresentation of the PacketExtension.

Static Public Attributes

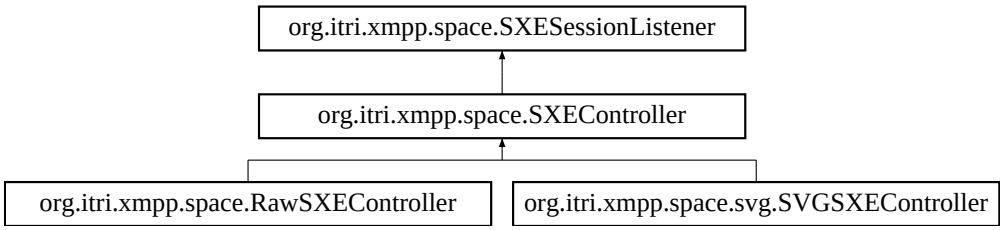
- static final String **ELEMENT_NAME** = "sxe"
- static final String **XMLNS** = "urn:xmpp:tmp:sxe"

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/packet/SXE.java

1.88 org.itri.xmpp.space.SXEController Class Reference

Abstract class for controlling an document in an SXE session. Inheritance diagram for org.itri.xmpp.space.SXEController::



Public Member Functions

- void initializeDocument (String localIdentifier)

  Initialize the document (add root element, set document type and head).

- Space getSpace ()

  Get the associated space session.

- SXESession getSXESession ()

  Get the associated SXE session.

- boolean handlesName ()

  Return true if this sxe session handles session name.

- String getName ()

  Return the name of this SXE session.

- boolean handlesInfo ()

  Return true if this sxe session handles session description.

- String getInfo ()

  Return the description of this sxe session.

- void documentUpdated ()

  Notification that the document has been updated.

- abstract void **parseDOM** () throws ClientException

Protected Member Functions

- **SXEController** (SXESession sxeSession, Space space)
- String **nextId** ()
- void addText (Element e, String text)

  Add text as sub content.

- void addAttribute (Element e, String attributeName, String value)

  Add an attribute.

- void changeAttribute (Element e, String attributeName, String newValue)

  Change or add the value of an attribute.

- void changeSubText (Element e, String text)

  Change the first sub text to the given string.

- void diffAndSend (Element e) throws ClientException

  Differentiate the Element from current and send the state changes to the space.

- abstract String **getDocumentType** ()
- abstract String **getDocumentHead** ()
- abstract Element **rootElement** ()

Protected Attributes

- SXESession **sxeSession**
- Space **space**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SXEController.java

1.89 org.itri.xmpp.space.SXEControllerFactory Interface Reference

Inherited by org.itri.xmpp.space.RawSXEController.RawSXEControllerFactory, and org.itri.xmpp.space.svg.SVGSXEController.SVGSXEControllerFactory.

Public Member Functions

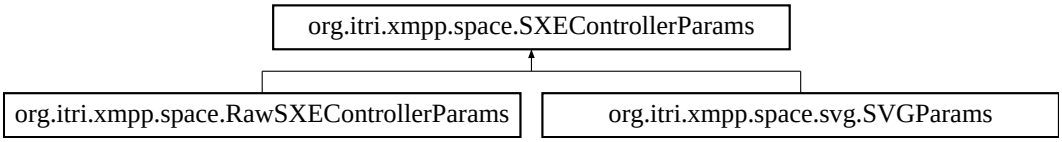
- SXEController **createSXEController** (SXESession sxeSession, Space space)

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/space/SXEControllerFactory.java

1.90 org.itri.xmpp.space.SXEControllerParams Interface Reference

Inheritance diagram for org.itri.xmpp.space.SXEControllerParams::



Public Member Functions

- SXEControllerFactory **getSXEControllerFactory** ()

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/space/SXEControllerParams.java

1.91 org.itri.xmpp.provider.SXEProvider Class Reference

Inherits org::jivesoftware::smack::provider::PacketExtensionProvider.

Public Member Functions

- PacketExtension **parseExtension** (XmlPullParser parser) throws Exception

Static Package Functions

- **[static initializer]**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/provider/SXEProvider.java

1.92 org.itri.xmpp.space.SXESession Class Reference

This class implements an experimental protocol for Shared XML Editing (SXE) initially proposed in http://xmpp.org/inbox/sxe.html .

Classes

- class Edit

  State change that changes an existing node.

- class EditAttr

  State change for changing an attribute node.

- class EditChdata

  State change that changes an existing node that contains character data.

- class EditComment

  State change for changing an comment node.

- class EditElement

  State change changin an element.

- class EditProcessingInstruction

  State change for changing a processing instruction node.

- class EditText

  State change for changing a text node.

- class New

  State change that creates a new node.

- class **OrphanedContent**

  Throwable thrown to indicate that a DOM node has become an orphan.

- interface StateChange

  Interface for state changes.

Public Member Functions

- SXESession (String id)

  Create a new SXE Session.

- void addStateListener (SXESessionStateListener listener)

  Add a state listener.

- void removeStateListener (SXESessionStateListener listener)

  Remove a state listener.

- void addSXESessionListener (SXESessionListener listener)

  Add an SXE session listener.

- void removeSXESessionListener (SXESessionListener listener)

  Remove SXE session listener.

- SXE getStateRequestPacket ()

  Get a SXE packet for requesting a state.

- synchronized SXE getState ()

  Get the current state of the session.

- String getDocumentType ()

  Get the document type / content type of this session.

- Document getDOM ()

  Get the document object model.

- void initialEmptyState (String from, String documentType, String documentHead)

  Initialize an empty state.

- synchronized Element getRootElement ()

  Get the root element.

- synchronized void addRootElement (Element element)

  Add root element.

- synchronized SXE elementToSXE (Element e)

  Convert a (new) element into state change instructions.

- synchronized SXE removeContent (Content c) throws ClientException

  Remove content.

- synchronized SXE diffElement (Element e) throws ClientException

  It's not allowed to have attributes "moved" from an element and then use this function because moved elements will be considered as removed.

- synchronized void **diffElementTo** (Element e, Element oe, SXE sxe) throws ClientException
- void setTestMode ()

  DEBUG ONLY.

- synchronized String nextContentId ()

  Generate a new ID used for a identifying a DOM node.

- synchronized void discardState ()

  Unset initial state putting the SXE session in a state where it waits for a new state.

- synchronized String synchronousPrivateMessage (Message message) throws ClientException

  Synchronous message call.

- synchronized void processMessage (Message message) throws ClientException

  Process an incoming group message that possibly may contain a SXE packet.

- synchronized String toXML ()

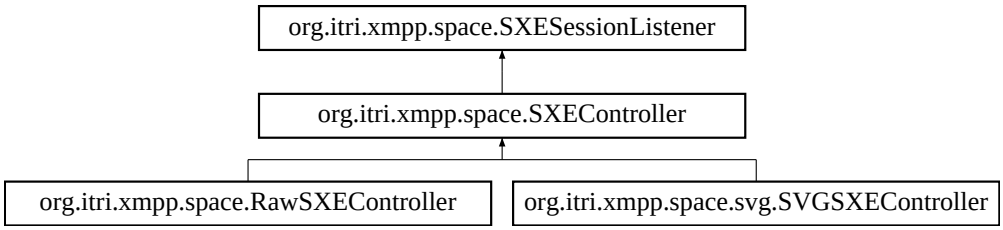
  Return the DOM Tree as XML output.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/SXESession.java

1.93 org.itri.xmpp.space.SXESessionListener Interface Reference

Notification about the DOM of a SXE Session has been changed. Inheritance diagram for org.itri.xmpp.space.SXESessionListener::



Public Member Functions

- void documentUpdated ()

  Notification that the document has been updated.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/space/SXESessionListener.java

1.94 org.itri.xmpp.space.SXESessionStateListener Interface Reference

Notification about updates of a Shared XML Editing Session state.

Inherited by org.itri.xmpp.space.JGroupsSpace.ActivateThread, and org.itri.xmpp.space.JGroupsSpace.StateMergeThread.

Public Member Functions

- void stateReceived ()

  Notification that the state has been received.

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/space/SXESessionStateListener.java

1.95 org.itri.xmpp.SystemProperties Class Reference

Static Public Member Functions

- static void setLocalHost (InetAddress newLocalHost)

  Set the local host address of this host.

- static InetAddress getLocalHost () throws ClientException

  Get the address of this host.

Static Public Attributes

- static final String **DEFAULT_INTERFACE** = "eth0"

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/SystemProperties.java

1.96 org.itri.xmpp.sample.TestUI Class Reference

Classes

- class **PListener**
- class **SListener**
- class **TestSpaceListener**

Public Member Functions

- void **contactAdded** (Contact contact)
- void **contactRemoved** (Contact contact)
- void **presenceUpdated** (Presence presence)
- void **run** ()

Static Public Member Functions

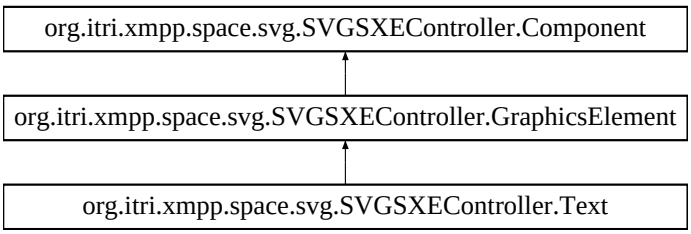
- static void **main** (String[ ] args)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/sample/TestUI.java

1.97 org.itri.xmpp.space.svg.SVGSXEController.Text Class Reference

SVG Text. Inheritance diagram for org.itri.xmpp.space.svg.SVGSXEController.Text::



Public Member Functions

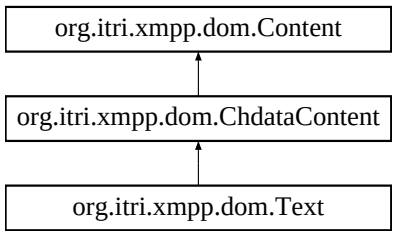
- **Text** (Element e) throws ClientException
- **Text** (float x, float y, String text, FontSize fontSize, SVGPaint p)
- float **getX** ()
- float **getY** ()
- FontSize **getFontSize** ()
- String **getText** ()
- void **move** (float x, float y) throws ClientException
- Element **generateShapeElement** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/space/svg/SVGSXEController.java

1.98 org.itri.xmpp.dom.Text Class Reference

XML Text node. Inheritance diagram for org.itri.xmpp.dom.Text::



Public Member Functions

- Text (String rid, String chdata)

  Create a new XML text node.

- String toXML ()

  Generate the escaped XML output of this text content.

Protected Member Functions

- Content _cloneContent ()

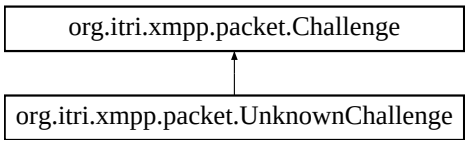
  Clone text content.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/dom/Text.java

1.99 org.itri.xmpp.packet.UnknownChallenge Class Reference

The challenge returned when no provider is found. Inheritance diagram for org.itri.xmpp.packet.UnknownChallenge::



Public Member Functions

- **UnknownChallenge** (String xmlns)
- String getElementName ()

  Get the element name which is "challenge".

- String getNamespace ()

  Get the namespace.

- String toXML ()

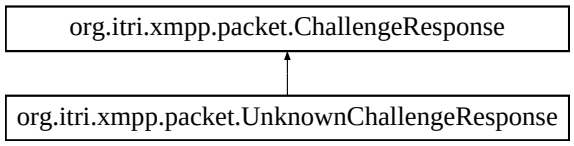
  Convert to XML string.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/packet/UnknownChallenge.java

1.100 org.itri.xmpp.packet.UnknownChallengeResponse Class Reference

The challenge response returned when no provider is found. Inheritance diagram for org.itri.xmpp.packet.UnknownChallengeResponse::



Public Member Functions

- **UnknownChallengeResponse** (String xmlns)
- String getElementName ()

  Get the element name which is "response".

- String getNamespace ()

  Get the namespace.

- String toXML ()

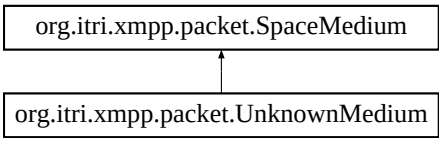
  Convert to XML string.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/packet/UnknownChallengeResponse.java

1.101 org.itri.xmpp.packet.UnknownMedium Class Reference

The medium returned when no provider is found. Inheritance diagram for org.itri.xmpp.packet.UnknownMedium::



Public Member Functions

- **UnknownMedium** (String xmlns)
- String getElementName ()

  Get the element name which is "medium".

- String getNamespace ()

  Get the namespace.

- String toXML ()

  Convert to XML string.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/packet/UnknownMedium.java

1.102 org.itri.xmpp.util.Utils Class Reference

Useful utilities.

Static Public Member Functions

- static char randomChar ()

  Generate a random character between a-z, A-Z and 0-9.

- static char randomDigit ()

  Generate a random digit between 0 and 9.

- static String randomString (int length)

  Generate a string consisting of random characters between a-z, A-Z and 0-9.

- static String randomDigitString (int length)

  Generate a string containing a specified number of random digits between 0 and 9.

Static Package Attributes

- static Random **random** = new Random()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/util/Utils.java

G Code Documentation: Android Interface

Space client (Android interface)

Generated by Doxygen 1.6.1

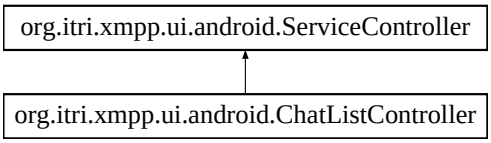
Sun Dec 6 15:10:20 2009

Contents

1 Class Documentation

1.1 org.itri.xmpp.ui.android.ChatListController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.ChatListController::



Classes

- class ChatViewAccessor

Public Member Functions

- **ChatListController** (SpaceClient client, ChatViewManager manager) throws ServiceException
- void **showChatListDialog** ()
- List< ChatViewAccessor > **getChatViewAccessors** ()
- void **userSelected** (ChatViewAccessor accessor)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/ChatListController.java

1.2 org.itri.xmpp.ui.android.ChatListDialog Class Reference

Inherits android::app::Dialog.

Public Member Functions

- **ChatListDialog** (Context context, ChatListController controller)

Protected Member Functions

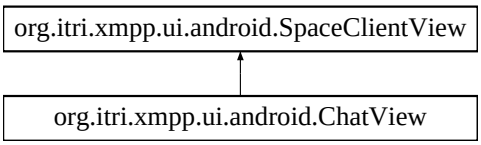
- void **userSelected** (int position)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/ChatListDialog.java

1.3 org.itri.xmpp.ui.android.ChatView Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.ChatView::



Public Member Functions

- **ChatView** (Context context, ChatViewController controller)
- View **getView** ()
- String **getTitle** ()
- void **sendButtonClicked** ()
- void **presenceUpdated** ()
- void **newMessage** (String from, String body)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/ChatView.java

1.4 org.itri.xmpp.ui.android.ChatListController.ChatViewAccessor Class Reference

Public Member Functions

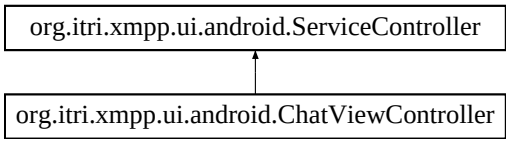
- String **getEntityId** ()
- String **getDisplayName** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/ChatListController.java

1.5 org.itri.xmpp.ui.android.ChatViewController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.ChatViewController::



Classes

- class **MessageSenderThread**

Public Member Functions

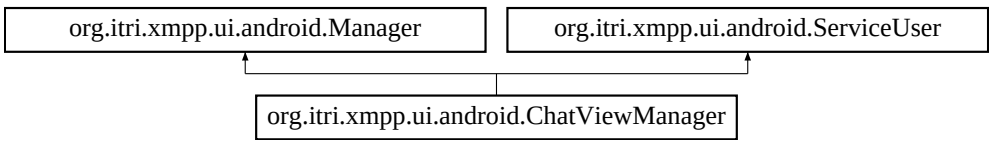
- **ChatViewController** (SpaceClient client, Chat chat) throws ServiceException
- String **getEntityId** ()
- String **getDisplayName** ()
- String **getDisplayName** (Presence presence)
- void **setClientView** ()
- int **getNumberOfUnread** ()
- String **getStatusMessage** ()
- Presence **getPresence** ()
- Session **getSession** ()
- synchronized void **presenceUpdated** (Presence p)
- void **sendMessage** (String s)
- void **newMessage** (Message m)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/ChatViewController.java

1.6 org.itri.xmpp.ui.android.ChatViewManager Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.ChatViewManager::



Public Member Functions

- **ChatViewManager** (SpaceClient client) throws ServiceException
- void **activateServiceUser** (SpaceClientService service)
- void **deactivateServiceUser** ()
- void **presenceUpdated** (Presence presence)
- List< ChatViewController > **getChatViewControllers** ()
- void **setClientViewTo** (Session session, String entity) throws ServiceException

- ChatViewController **getChatController** (Session session, String entity) throws ServiceException

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/ChatViewManager.java

1.7 org.itri.xmpp.ui.android.ColorPickerController.ColorChangedListener Interface Reference

Public Member Functions

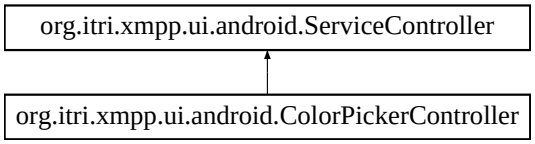
- void **colorChanged** (int color)

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/ui/android/ColorPickerController.java

1.8 org.itri.xmpp.ui.android.ColorPickerController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.ColorPickerController::



Classes

- interface ColorChangedListener

Public Member Functions

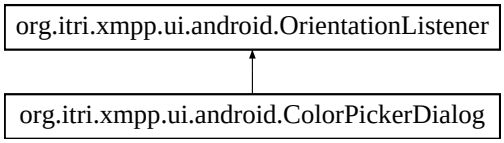
- **ColorPickerController** (SpaceClient client, ColorChangedListener listener, int initialColor)
- void **showColorPickerDialog** (Color initialColor)
- int **getInitialColor** ()
- void **colorChanged** (int newColor)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/ColorPickerController.java

1.9 org.itri.xmpp.ui.android.ColorPickerDialog Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.ColorPickerDialog::



Classes

- class **ColorPickerView**
- interface **OnColorPreviewChangedListener**

Public Member Functions

- **ColorPickerDialog** (Context context, ColorPickerController controller)
- void **orientationChanged** ()
- void **onProgressChanged** (SeekBar seekBar, int progress, boolean fromUser)
- void **onStartTrackingTouch** (SeekBar seekBar)
- void **onStopTrackingTouch** (SeekBar seekBar)

Protected Member Functions

- void **onCreate** (Bundle savedInstanceState)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/ColorPickerDialog.java

1.10 org.itri.xmpp.ui.android.ColorPreviewView Class Reference

Inherits android::view::View.

Public Member Functions

- **ColorPreviewView** (Context context, SVGController controller)
- void **onDraw** (Canvas canvas)

Protected Member Functions

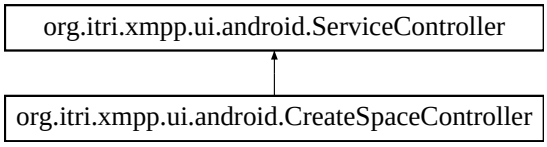
- void **onMeasure** (int widthMeasureSpec, int heightMeasureSpec)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/ColorPreviewView.java

1.11 org.itri.xmpp.ui.android.CreateSpaceController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.CreateSpaceController::



Public Member Functions

- **CreateSpaceController** (SpaceClient client) throws ServiceException
- void **showCreateSpaceDialog** ()
- void createSpace (String name, SpaceClientService.SpaceMediumType mediumType, SpaceClientService.SpaceContentType contentType, String password)

  Show a progress dialog, then ask the service to create a space.

- String getString (int rid)

  Get the string given the resource id.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/CreateSpaceController.java

1.12 org.itri.xmpp.ui.android.CreateSpaceDialog Class Reference

Inherits android::app::Dialog.

Public Member Functions

- **CreateSpaceDialog** (Context context, CreateSpaceController controller)

Protected Member Functions

- void **onStart** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/CreateSpaceDialog.java

1.13 org.itri.xmpp.ui.android.FunCall Class Reference

Public Member Functions

- **FunCall** (Object o)
- abstract void **run** ()

Protected Attributes

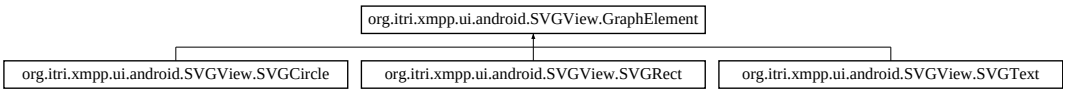
- Object **_o**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/FunCall.java

1.14 org.itri.xmpp.ui.android.SVGView.GraphElement Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SVGView.GraphElement::



Public Member Functions

- String **getRid** ()
- SVGViewPaint **getSVGViewPaint** ()
- SVGPaint **getSVGPaint** ()
- SVGSXEController.GraphicsElement **getGraphicsElement** ()

Protected Member Functions

- **GraphElement** (SVGViewPaint paint)
- **GraphElement** (SVGSXEController.GraphicsElement ge)
- abstract SVGSXEController.GraphicsElement **generateGraphicsElement** ()

Protected Attributes

- SVGViewPaint **svgViewPaint**
- SVGSXEController.GraphicsElement **ge**

Package Functions

- abstract boolean **isWithin** (int x, int y)
- abstract ResizeAction.Point **getPoint** (int x, int y)
- abstract int **getOuterFromX** ()
- abstract int **getOuterFromY** ()
- abstract int **getOuterToX** ()
- abstract int **getOuterToY** ()
- abstract int **getX** ()
- abstract int **getY** ()
- abstract int **getWidth** ()
- abstract int **getHeight** ()
- abstract Path **getScaledPath** (int xdiff, int ydiff)
- abstract Path **getPath** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGView.java

1.15 org.itri.xmpp.ui.android.ItemSelectController.ItemAccessor< ItemType > Interface Reference

Public Member Functions

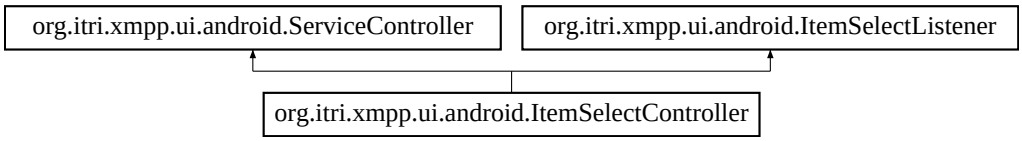
- String **getDisplayName** ()
- ItemType **getItem** ()

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/ui/android/ItemSelectController.java

1.16 org.itri.xmpp.ui.android.ItemSelectController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.ItemSelectController::



Classes

- interface ItemAccessor< ItemType >

Public Member Functions

- **ItemSelectController** (SpaceClient client) throws ServiceException
- void **showItemSelectDialog** ()
- abstract List< ItemAccessor > **getItemAccessors** ()
- void **itemSelected** (ItemAccessor accessor)

Protected Member Functions

- abstract void **onItemSelected** (ItemAccessor accessor)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/ItemSelectController.java

1.17 org.itri.xmpp.ui.android.ItemSelectDialog Class Reference

Inherits android::app::Dialog.

Public Member Functions

- **ItemSelectDialog** (Context context, String title, List< String > items, ItemSelectListener listener)

Protected Member Functions

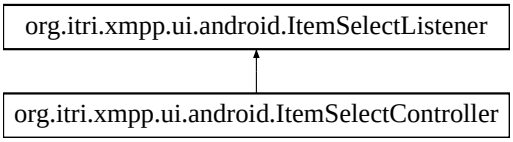
- void **itemSelected** (int position)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/ItemSelectDialog.java

1.18 org.itri.xmpp.ui.android.ItemSelectListener Interface Reference

Inheritance diagram for org.itri.xmpp.ui.android.ItemSelectListener::



Public Member Functions

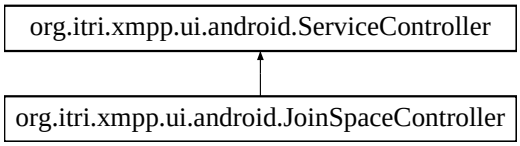
- void **itemSelected** (int position)

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/ui/android/ItemSelectListener.java

1.19 org.itri.xmpp.ui.android.JoinSpaceController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.JoinSpaceController::



Classes

- class **PasswordFormDialog**

Public Member Functions

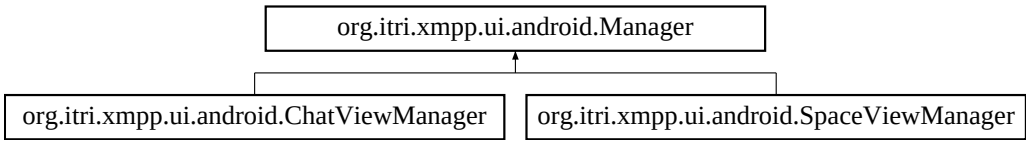
- **JoinSpaceController** (SpaceClient client) throws ServiceException
- void **joinSpace** (String uuid, String contact)
- void **joinFailed** (JoinSpace js, String msg, Exception e)
- void **joinFailed** (JoinSpace js, String msg)
- void **joinFinished** (JoinSpace js, Space space)
- void **challenge** (JoinSpace js, Challenge challenge)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/JoinSpaceController.java

1.20 org.itri.xmpp.ui.android.Manager Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.Manager::



Protected Member Functions

- **Manager** (SpaceClient client)
- Object getObject ()

  Get object from service.

- void **storeObject** (Object o)

Protected Attributes

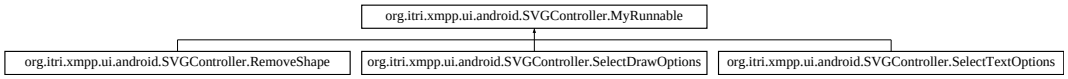
- SpaceClient **client**
- SpaceClientService **service**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/Manager.java

1.21 org.itri.xmpp.ui.android.SVGController.MyRunnable Interface Reference

Inheritance diagram for org.itri.xmpp.ui.android.SVGController.MyRunnable::



Public Member Functions

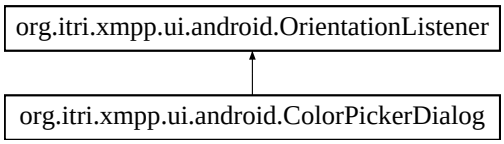
- void **run** () throws ClientException

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGController.java

1.22 org.itri.xmpp.ui.android.OrientationListener Interface Reference

Inheritance diagram for org.itri.xmpp.ui.android.OrientationListener::



Public Member Functions

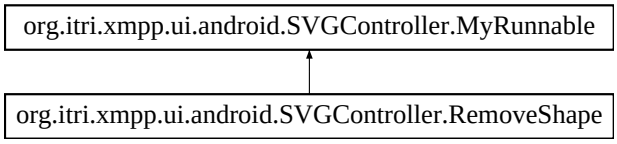
- void **orientationChanged** ()

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/ui/android/OrientationListener.java

1.23 org.itri.xmpp.ui.android.SVGController.RemoveShape Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SVGController.RemoveShape::



Public Member Functions

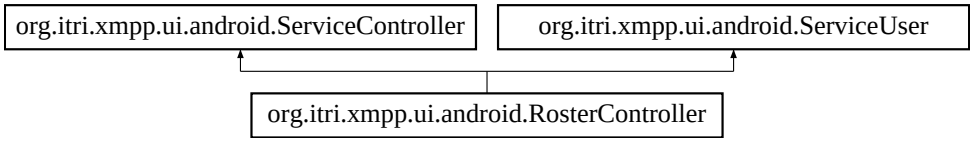
- void **run** () throws ClientException

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGController.java

1.24 org.itri.xmpp.ui.android.RosterController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.RosterController::



Public Member Functions

- **RosterController** (SpaceClient client) throws ServiceException
- void **setSpaceView** ()
- void **activateServiceUser** (SpaceClientService service)
- void **deactivateServiceUser** ()
- void **openChatFor** (String contact)
- void presenceUpdated (Presence presence)

  Called to uptade view of presence.

- void **serviceNameChanged** (String newName, String oldName)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/RosterController.java

1.25 org.itri.xmpp.ui.android.RosterItemView Class Reference

Inherits android::widget::LinearLayout.

Classes

- class **KeyValueView**

Public Member Functions

- String **toString** ()
- **RosterItemView** (Context context, String entityName)
- void **updatePresence** (Presence presence)
- void toggleExpanded ()

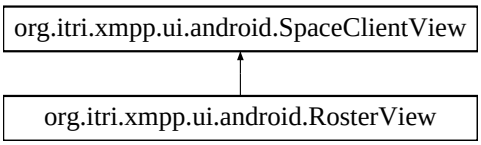
  Convenience method to expand or hide the dialogue.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/RosterItemView.java

1.26 org.itri.xmpp.ui.android.RosterView Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.RosterView::



Classes

- class **RosterAdapter**

  A sample ListAdapter that presents content from arrays of speeches and text.

Public Member Functions

- **RosterView** (Context context, RosterController controller)
- View **getView** ()
- String **getTitle** ()
- void presenceUpdated (Presence presence)

  Called to uptade view of presence.

- void serviceNameChanged (String newName, String oldName)

  Called when Link-local service name changed.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/RosterView.java

1.27 org.itri.xmpp.ui.android.SpaceClientService.SCBinder Class Reference

Inherits android::os::Binder.

Package Functions

- SpaceClientService **getService** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SpaceClientService.java

1.28 org.itri.xmpp.ui.android.SearchSpaceDialog.SearchResultView Class Reference

Inherits android::widget::LinearLayout.

Public Member Functions

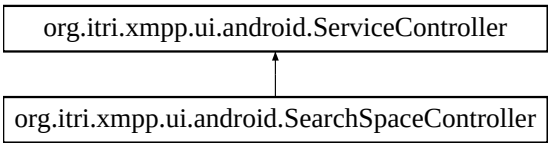
- **SearchResultView** (Context context, SpaceInfo si)
- SpaceInfo **getSpaceInfo** ()
- void **newResult** (SpaceInfo spaceInfo, String newEntity)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SearchSpaceDialog.java

1.29 org.itri.xmpp.ui.android.SearchSpaceController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SearchSpaceController::



Public Member Functions

- **SearchSpaceController** (SpaceClient client) throws ServiceException
- void **showSearchSpaceDialog** ()
- void **startSpaceSearch** ()
- void **searchEnded** (boolean success)
- void **spaceSelected** (SpaceInfo spaceInfo)
- String **contactToName** (String contact)
- synchronized void **spaceDiscovered** (SpaceInfo spaceInfo)
- void **spaceContactAdded** (SpaceInfo spaceInfo, String entityID)
- void **coughtException** (ClientException ce)
- synchronized void **searchAborted** (ClientException ce)
- synchronized void **searchFinished** ()
- boolean **stringEnough** (String str)

Package Functions

- void **joinSpace** (String uuid, String contact)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SearchSpaceController.java

1.30 org.itri.xmpp.ui.android.SearchSpaceDialog Class Reference

Inherits android::app::Dialog.

Classes

- class **SearchResultAdapter**
- class SearchResultView

Public Member Functions

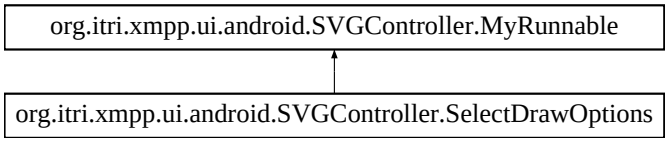
- **SearchSpaceDialog** (Context context, SearchSpaceController controller)
- void **prepareSearch** ()
- void **searchEnded** (boolean success)
- void **addResult** (SpaceInfo spaceInfo)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SearchSpaceDialog.java

1.31 org.itri.xmpp.ui.android.SVGController.SelectDrawOptions Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SVGController.SelectDrawOptions::



Public Member Functions

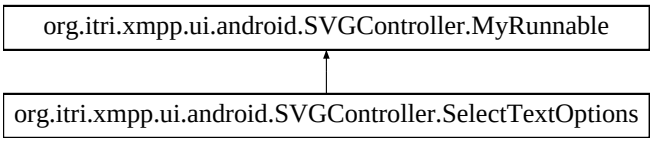
- void **run** () throws ClientException

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGController.java

1.32 org.itri.xmpp.ui.android.SVGController.SelectTextOptions Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SVGController.SelectTextOptions::



Public Member Functions

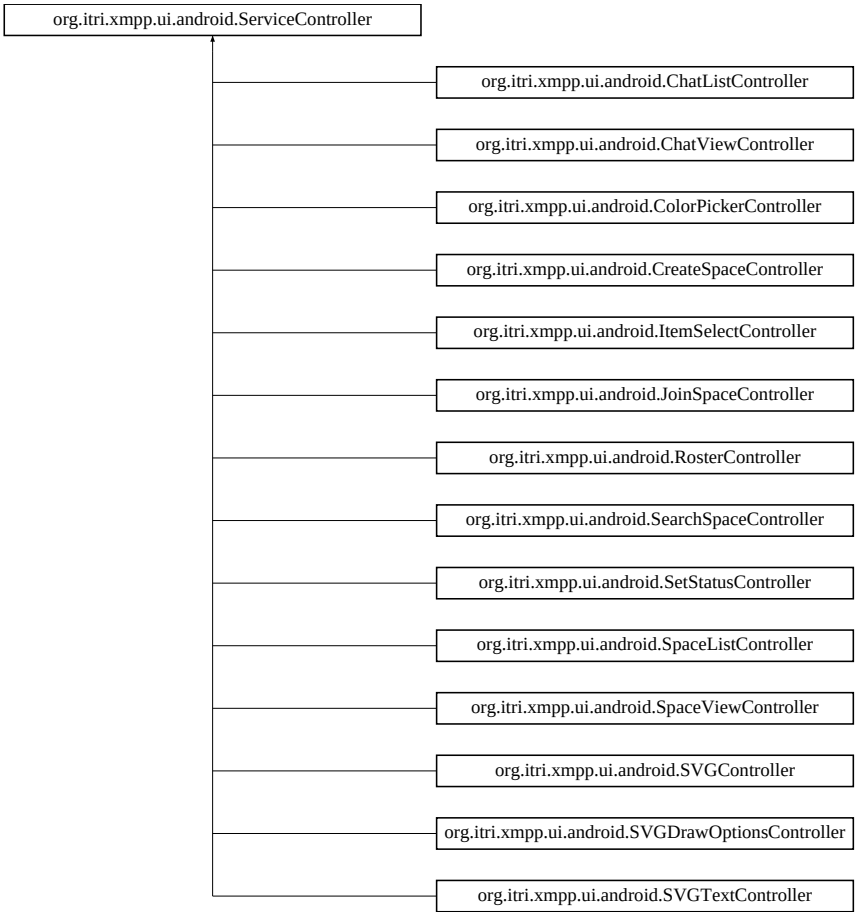
- void **run** () throws ClientException

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGController.java

1.33 org.itri.xmpp.ui.android.ServiceController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.ServiceController::



Public Member Functions

- void remoteRun (Runnable runnable)

  Run the given runnable on the UI thread.

- String getString (int rid)

  Get the string given the resource id.

- SpaceClient **getClient** ()
- void setProgressVisibility (boolean visible)

  Set progress marker visibility.

- void **addOrientationListener** (OrientationListener listener)
- boolean **isHorizontal** ()

Protected Member Functions

- **ServiceController** (SpaceClient client)

- boolean isServiceDependent ()

  Returns true if the controller is dependent on the service.

Protected Attributes

- SpaceClient **client**
- SpaceClientService **service**

Package Functions

- void **alertDialog** (String title, String message)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/ServiceController.java

1.34 org.itri.xmpp.ui.android.ServiceException Class Reference

Public Member Functions

- **ServiceException** (int reason)
- **ServiceException** (Throwable cause)
- **ServiceException** (Throwable cause, int reason)
- int **getReasonId** ()
- int **toErrorMessageId** ()

Static Public Attributes

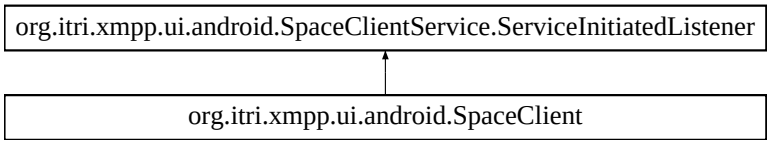
- static final int **UNKNOWN_REASON** = 0
- static final int **NO_SERVICE** = 0x0a01
- static final int **NO_LLSESSION** = 0x0b01
- static final int **SPACE_UNKNOWN_TYPE** = 0x0f01
- static final int **SPACE_UNKNOWN_MEDIUM** = 0x0f02
- static final int **SPACE_JGROUPS_REQUIRE_LLSESSION** = 0x0f03
- static final int **SPACE_SEARCH_REQUIRE_LLSESSION** = 0x0e01
- static final int **SPACE_SEARCH_MULTIPLE** = 0x0e02

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/ServiceException.java

1.35 org.itri.xmpp.ui.android.SpaceClientService.ServiceInitiatedListener Interface Reference

Inheritance diagram for org.itri.xmpp.ui.android.SpaceClientService.ServiceInitiatedListener::



Public Member Functions

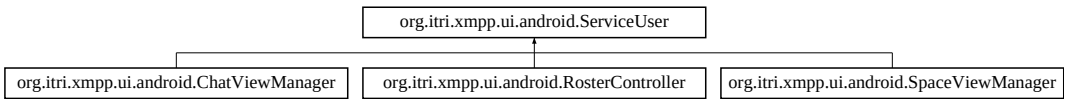
- void **spaceClientServiceInitiated** (boolean success)

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/ui/android/SpaceClientService.java

1.36 org.itri.xmpp.ui.android.ServiceUser Interface Reference

Inheritance diagram for org.itri.xmpp.ui.android.ServiceUser::



Public Member Functions

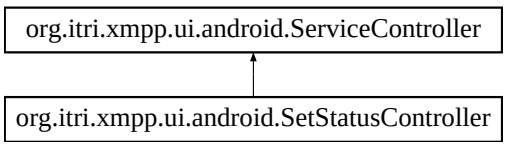
- void **activateServiceUser** (SpaceClientService service)
- void **deactivateServiceUser** ()

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/ui/android/ServiceUser.java

1.37 org.itri.xmpp.ui.android.SetStatusController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SetStatusController::



Public Member Functions

- **SetStatusController** (SpaceClient client) throws ServiceException
- void **showSetStatusDialog** ()

Package Functions

- void **setStatus** (Presence.Mode mode, String message)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SetStatusController.java

1.38 org.itri.xmpp.ui.android.SetStatusDialog Class Reference

Inherits android::app::Dialog.

Public Member Functions

- **SetStatusDialog** (Context context, SetStatusController controller)

Protected Member Functions

- void **onStart** ()

Package Functions

- void setCurrentModeAndMessage (Presence.Mode mode, String message)

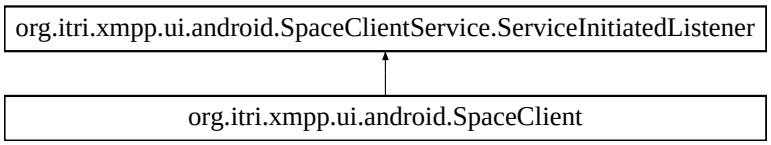
  Set current presence mode value and message value as default values of this dialog.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SetStatusDialog.java

1.39 org.itri.xmpp.ui.android.SpaceClient Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SpaceClient::



Classes

- class **SpaceClientServiceConnection**
- class **UpdateDoer**

Public Member Functions

- SpaceClientService **getService** ()
- void **addServiceUser** (ServiceUser newServiceUser)
- void **addOrientationListener** (OrientationListener listener)
- void **spaceClientServiceInitiated** (boolean success)
- void **onCreate** (Bundle savedInstanceState)
- void **onStart** ()
- boolean **onKeyDown** (int keyCode, KeyEvent event)
- void **onConfigurationChanged** (Configuration newConfiguration)
- void **onStop** ()
- void **onDestroy** ()
- boolean **onCreateOptionsMenu** (Menu menu)
- boolean **onOptionsItemSelected** (MenuItem item)
- void **spam** ()

Package Functions

- void **setClientView** (SpaceClientView view)
- void **remoteRun** (Runnable doer)
- boolean **isHorizontal** ()
- void **alertDialog** (String title, String message)
- String **errorMessage** (ServiceException se)

Package Attributes

- SpaceViewManager **spaceViewManager**
- ChatViewManager **chatViewManager**
- RosterController **rosterController**
- SetStatusController **setStatusController**

- CreateSpaceController **createSpaceController**
- SearchSpaceController **searchSpaceController**
- JoinSpaceController **joinSpaceController**
- ChatListController **chatListController**
- SpaceListController **spaceListController**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SpaceClient.java

1.40 org.itri.xmpp.ui.android.SpaceClientService Class Reference

Inherits android::app::Service, and org::itri::xmpp::SessionListener.

Classes

- class **InitSessions**
- class SCBinder
- interface ServiceInitiatedListener
- class WifiStateListener

Public Member Functions

- void **onCreate** ()
- void **onStart** (Intent intent, int startId)
- IBinder **onBind** (Intent intent)
- void **onDestroy** ()
- void **storeObject** (Class clazz, Object object)
- Object **getStoredObject** (Class clazz)
- Collection< Session > **getAllSessions** ()
- SessionManager **getSM** ()
- LinkLocalSession **getLLSession** () throws ServiceException
- void **addSessionListener** (SessionListener listener)
- void **removeSessionListener** (SessionListener listener)
- void **newContactListener** (ContactListener listener)
- void **newPresenceListener** (PresenceListener listener)
- void **newLinkLocalStateListener** (LinkLocalStateListener listener)
- void **removeContactListener** (ContactListener listener)
- void **removePresenceListener** (PresenceListener listener)
- void **removeLinkLocalStateListener** (LinkLocalStateListener listener)
- void **sessionActivating** (Session session)
- void **sessionActivated** (Session session)
- void **sessionDeactivated** (Session session)
- void **sessionActivationFailed** (Session session, ClientException ce)
- void **setServiceInitiatedListener** (ServiceInitiatedListener listener)

- boolean **isInitiated** ()
- void **initiate** ()
- void **quit** ()
- LinkLocalSession **getLinkLocalSession** ()

Package Types

- enum **SpaceMediumType** { **jgroups** }
- enum **SpaceContentType** { **svg** }

Package Functions

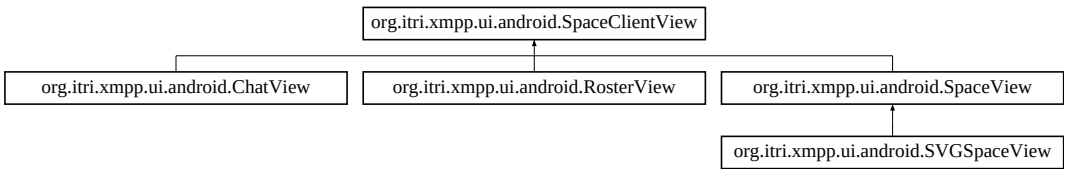
- boolean **hasActiveSession** ()
- boolean **hasActiveLinkLocalSession** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SpaceClientService.java

1.41 org.itri.xmpp.ui.android.SpaceClientView Interface Reference

Inheritance diagram for org.itri.xmpp.ui.android.SpaceClientView::



Public Member Functions

- View **getView** ()
- String **getTitle** ()

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/ui/android/SpaceClientView.java

1.42 org.itri.xmpp.ui.android.SpaceInfoDialog Class Reference

Inherits android::app::Dialog.

Public Member Functions

- **SpaceInfoDialog** (Context context, SearchSpaceController controller, SpaceInfo spaceInfo)

Protected Member Functions

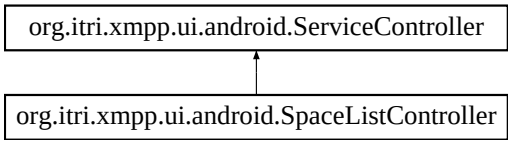
- void **onStart** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SpaceInfoDialog.java

1.43 org.itri.xmpp.ui.android.SpaceListController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SpaceListController::



Classes

- class SpaceViewAccessor

Public Member Functions

- **SpaceListController** (SpaceClient client, SpaceViewManager manager) throws ServiceException
- void **showSpaceListDialog** ()
- List< SpaceViewAccessor > **getSpaceViewAccessors** ()
- void **spaceSelected** (SpaceViewAccessor accessor)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SpaceListController.java

1.44 org.itri.xmpp.ui.android.SpaceListDialog Class Reference

Inherits android::app::Dialog.

Public Member Functions

- **SpaceListDialog** (Context context, SpaceListController controller)

Protected Member Functions

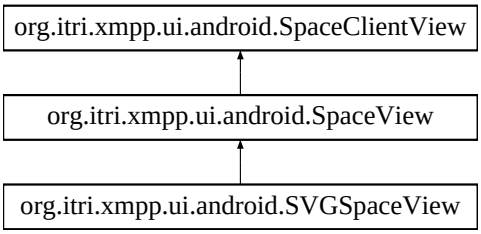
- void **spaceSelected** (int position)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SpaceListDialog.java

1.45 org.itri.xmpp.ui.android.SpaceView Interface Reference

Inheritance diagram for org.itri.xmpp.ui.android.SpaceView::



Public Member Functions

- void **documentUpdated** ()

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/ui/android/SpaceView.java

1.46 org.itri.xmpp.ui.android.SpaceListController.SpaceViewAccessor Class Reference

Public Member Functions

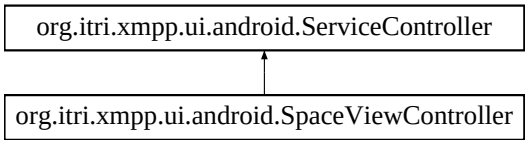
- String **getUUID** ()
- String **getSpaceName** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SpaceListController.java

1.47 org.itri.xmpp.ui.android.SpaceViewController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SpaceViewController::



Public Member Functions

- **SpaceViewController** (SpaceClient client, Space space, ProgressDialog loadingDialog) throws ServiceException
- void **spaceInitiated** ()
- void **setClientView** ()
- void **spaceInitiationFailed** (ClientException ce)
- void **spaceCrashed** (ClientException ce)
- void **spaceContentInvalid** (ClientException ce)
- void **spaceClosed** ()
- void **documentUpdated** ()
- String **getUUID** ()
- String **getSpaceName** ()
- String getString (int rid)

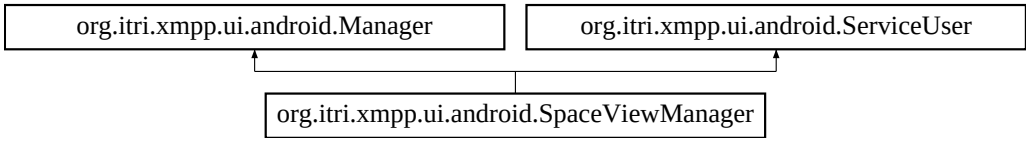
  Get the string given the resource id.

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SpaceViewController.java

1.48 org.itri.xmpp.ui.android.SpaceViewManager Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SpaceViewManager::



Public Member Functions

- **SpaceViewManager** (SpaceClient client) throws ServiceException
- void **activateServiceUser** (SpaceClientService service)
- void **deactivateServiceUser** ()
- List< SpaceViewController > **getSpaceViewControllers** ()
- void **newSpace** (Space space, ProgressDialog loadingDialog) throws ServiceException
- void **newSpace** (Space space)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SpaceViewManager.java

1.49 org.itri.xmpp.ui.android.SpinnerList< T > Class Reference

List intended for use with a android.widget.Spinner.

Inherits java::util::LinkedList< String >.

Public Member Functions

- void addSpinnerItem (String text, T object)

  Add a string with a corresponding object associated with it.

- T getCorrespondingAt (int position)

  Return the object associated with the position.

Package Attributes

- Map< Integer, T > **objectMap** = new HashMap<Integer,T>()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SpinnerList.java

1.50 org.itri.xmpp.ui.android.StatusIcon Class Reference

Inherits android::widget::ImageSwitcher.

Public Member Functions

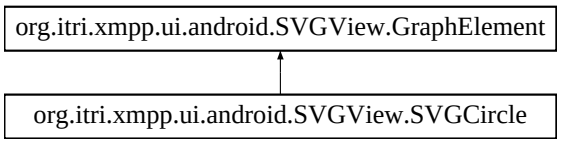
- **StatusIcon** (Context context)
- **StatusIcon** (Context context, AttributeSet attrs)
- void **setPresence** (Presence presence)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/StatusIcon.java

1.51 org.itri.xmpp.ui.android.SVGView.SVGCircle Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SVGView.SVGCircle::



Public Member Functions

- **SVGCircle** (int fromx, int fromy, int tox, int toy, SVGViewPaint paint)
- boolean **isWithin** (int xpos, int ypos)
- int **getX** ()
- int **getY** ()
- int **getOuterFromX** ()
- int **getOuterToX** ()
- int **getOuterFromY** ()
- int **getOuterToY** ()
- int **getHeight** ()
- int **getWidth** ()
- float **getR** ()
- ResizeAction.Point **getPoint** (int x, int y)
- Path **getScaledPath** (int xdiff, int ydiff)
- Path **getPath** ()

Protected Member Functions

- SVGSXEController.GraphicsElement **generateGraphicsElement** ()

Package Attributes

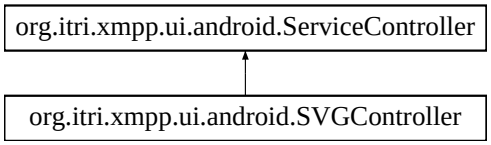
- int **x**
- int **y**
- float **r**
- float **svgx**
- float **svgy**
- float **svgr**

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGView.java

1.52 org.itri.xmpp.ui.android.SVGController Class Reference

Class used for controlling a SVG space view, such as selecting shapes and tools. Inheritance diagram for org.itri.xmpp.ui.android.SVGController::



Classes

- interface MyRunnable
- class RemoveShape
- class SelectDrawOptions
- class SelectTextOptions

Public Types

- enum **Shape** { **rectangle**, **circle**, **text** }
- enum **Tool** {
  draw, **select**, **move**, **resize**,
  color }
- enum **DrawMode** { **fill**, **stroke_fill**, **stroke** }

Public Member Functions

- **SVGController** (SpaceClient client, SVGSXEController sxe)
- SVGView **getSVGView** ()
- SVGSXEController **getSVGSXEController** ()

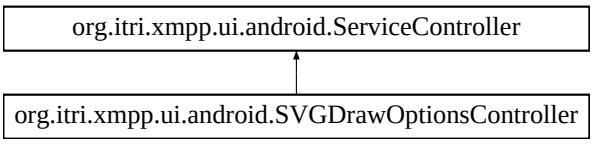
- SVGViewPaint **getSelectedOrDefaultSVGViewPaint** ()
- void **addOptionsListener** (SVGOptionsListener listener)
- void **removeOptionsListener** (SVGOptionsListener listener)
- void **notifySVGPaintChanged** ()
- void **notifyToolSelected** ()
- void **selectShape** ()
- void **selectTool** ()
- void **selectOptions** ()
- void **selectColor** ()
- void **viewTextOptions** ()
- void **viewOptions** ()
- void **enterSVGText** (SVGText svgText)
- void **newGraphicsElement** (GraphElement ge) throws ClientException
- void **newText** (SVGText svgText, String text, FontSize fontSize)
- void **drawModeSelected** (DrawMode dm, float strokeWidth)
- Shape **getShape** ()
- Tool **getTool** ()
- SVGPaint **getSVGPaint** ()
- SVGViewPaint **getSVGViewPaint** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGController.java

1.53 org.itri.xmpp.ui.android.SVGDrawOptionsController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SVGDrawOptionsController::



Classes

- interface SVGDrawOptionsListener

Public Member Functions

- **SVGDrawOptionsController** (SpaceClient client, SVGController controller)
- void **showSVGDrawOptionsDialog** (SVGViewPaint svgPaint)
- float **getStrokeWidth** ()
- DrawMode **getDrawMode** ()

- void **setDrawMode** (DrawMode dm, float strokeWidth)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGDrawOptionsController.java

1.54 org.itri.xmpp.ui.android.SVGDrawOptionsDialog Class Reference

Inherits android::app::Dialog.

Public Member Functions

- **SVGDrawOptionsDialog** (Context context, SVGDrawOptionsController controller)
- void **onClick** (View v)
- void **onProgressChanged** (SeekBar seekBar, int progress, boolean fromUser)
- void **onStartTrackingTouch** (SeekBar seekBar)
- void **onStopTrackingTouch** (SeekBar seekBar)

Protected Member Functions

- void **onCreate** (Bundle savedInstanceState)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGDrawOptionsDialog.java

1.55 org.itri.xmpp.ui.android.SVGDrawOptionsController.SVGDrawOptionsListener Interface Reference

Public Member Functions

- void **strokeWidthSelected** (int strokeWidth)

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGDrawOptionsController.java

1.56 org.itri.xmpp.ui.android.SVGView.SVGFont Class Reference

Public Member Functions

- **SVGFont** (SVGFontSize fs)

- SVGFontSize **getSVGFontSize** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGView.java

1.57 org.itri.xmpp.ui.android.SVGView.SVGFontSize Class Reference

Public Member Functions

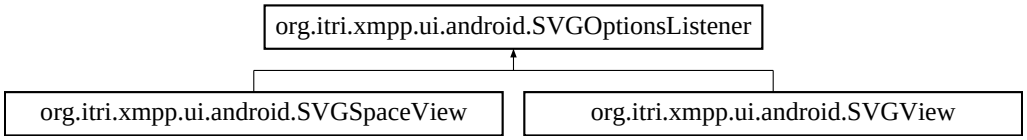
- **SVGFontSize** (FontSize fs)
- FontSize **getFontSize** ()
- float **getSize** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGView.java

1.58 org.itri.xmpp.ui.android.SVGOptionsListener Interface Reference

Inheritance diagram for org.itri.xmpp.ui.android.SVGOptionsListener::



Public Member Functions

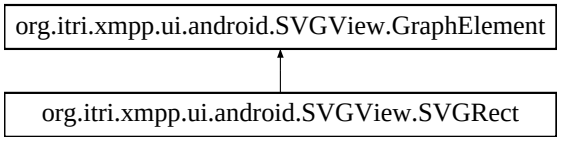
- void **svgPaintChanged** (SVGViewPaint paint)
- void **toolSelected** (SVGController.Tool tool)
- void **shapeSelected** (SVGController.Shape shape)

The documentation for this interface was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGOptionsListener.java

1.59 org.itri.xmpp.ui.android.SVGView.SVGRect Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SVGView.SVGRect::



Public Member Functions

- boolean **isWithin** (int x, int y)
- ResizeAction.Point **getPoint** (int x, int y)
- int **getX** ()
- int **getY** ()
- int **getWidth** ()
- int **getHeight** ()
- int **getOuterFromX** ()
- int **getOuterFromY** ()
- int **getOuterToX** ()
- int **getOuterToY** ()
- Path **getScaledPath** (int xdiff, int ydiff)
- Path **getPath** ()

Protected Member Functions

- SVGSXEController.GraphicsElement **generateGraphicsElement** ()

Package Attributes

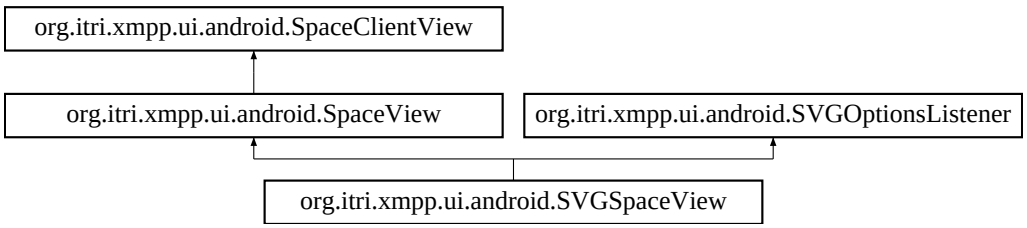
- int **fromx**
- int **fromy**
- int **tox**
- int **toy**
- int **maxx**
- int **maxy**
- int **minx**
- int **miny**
- float **svgx**
- float **svgy**
- float **svgwidth**
- float **svgheight**
- SVGViewPaint **paint** = null

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGView.java

1.60 org.itri.xmpp.ui.android.SVGSpaceView Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SVGSpaceView::



Public Member Functions

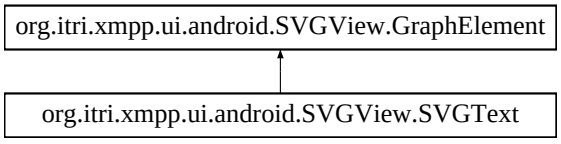
- **SVGSpaceView** (Context context, SpaceViewController controller, SVGController svgController)
- void **svgPaintChanged** (SVGViewPaint paint)
- void **toolSelected** (SVGController.Tool tool)
- void **shapeSelected** (SVGController.Shape shape)
- View **getView** ()
- String **getTitle** ()
- void **setVertical** ()
- void **setHorizontal** ()
- void **documentUpdated** ()
- void **onClick** (View view)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGSpaceView.java

1.61 org.itri.xmpp.ui.android.SVGView.SVGText Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SVGView.SVGText::



Public Member Functions

- **SVGText** (int fromx, int fromy, int tox, int toy, SVGFont font, SVGViewPaint paint)

- void **setFontSize** (FontSize fs)
- SVGFontSize **getSVGFontSize** ()
- void **setText** (String text)
- String **getText** ()
- boolean **isWithin** (int x, int y)
- int **getX** ()
- int **getY** ()
- int **getOuterFromX** ()
- int **getOuterToX** ()
- int **getOuterFromY** ()
- int **getOuterToY** ()
- int **getHeight** ()
- int **getWidth** ()
- ResizeAction.Point **getPoint** (int x, int y)
- Path **getScaledPath** (int xdiff, int ydiff)
- Path **getPath** ()

Protected Member Functions

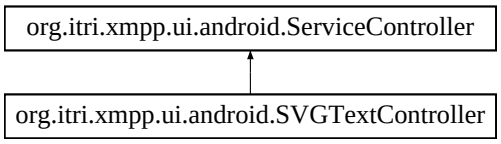
- SVGSXEController.GraphicsElement **generateGraphicsElement** ()

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGView.java

1.62 org.itri.xmpp.ui.android.SVGTextController Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SVGTextController::



Public Member Functions

- **SVGTextController** (SpaceClient client, SVGController controller)
- String **getCurrentText** ()
- FontSize **getCurrentFontSize** ()
- void **newText** (String text, FontSize fontSize)
- void **enterNewText** (SVGView.SVGText svgText)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGTextController.java

1.63 org.itri.xmpp.ui.android.SVGTextDialog Class Reference

Inherits android::app::Dialog.

Classes

- class **SelectSize**

Public Member Functions

- **SVGTextDialog** (Context context, SVGTextController controller)
- void **onClick** (View v)
- void **onProgressChanged** (SeekBar seekBar, int progress, boolean fromUser)
- void **onStartTrackingTouch** (SeekBar seekBar)
- void **onStopTrackingTouch** (SeekBar seekBar)

Protected Member Functions

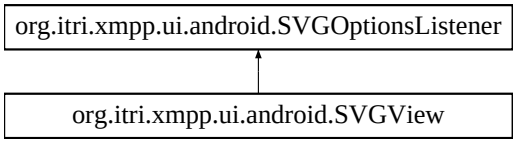
- void **onCreate** (Bundle savedInstanceState)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGTextDialog.java

1.64 org.itri.xmpp.ui.android.SVGView Class Reference

Inheritance diagram for org.itri.xmpp.ui.android.SVGView::



Classes

- class GraphElement
- class **ResizeAction**
- class SVGCircle
- class SVGFont
- class SVGFontSize
- class SVGRect
- class SVGText
- class **SVGViewPaint**

Public Member Functions

- **SVGView** (Context context, SVGController controller, SVGSXEController svg)

- GraphElement **getSelected** ()
- void **updateDrawables** ()
- void **toolSelected** (Tool tool)
- void **svgPaintChanged** (SVGViewPaint paint)
- void **shapeSelected** (SVGController.Shape shape)
- boolean **onTouchEvent** (MotionEvent me)

Protected Member Functions

- synchronized void **onDraw** (Canvas canvas)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SVGView.java

1.65 org.itri.xmpp.ui.android.UIUtils Class Reference

Static Public Member Functions

- static String **llPresenceToDisplayName** (LinkLocalPresence presence)
- static boolean **stringEnough** (String str)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/UIUtils.java

1.66 org.itri.xmpp.ui.android.SpaceClientService.WifiStateListener Class Reference

Inherits android::content::BroadcastReceiver.

Public Member Functions

- void **onReceive** (Context context, Intent intent)

The documentation for this class was generated from the following file:

- src/org/itri/xmpp/ui/android/SpaceClientService.java