

Themenkreis TK III: Anwendungsarchitekturen und Dienste

Erstautor: Frank Eyermann  
Universität der Bundeswehr München  
Institut für Technische Informatik  
Werner-Heisenberg-Weg 39  
85577 Neubiberg  
Tel.: 089-6004-2404  
frank.eyermann@unibw.de

Vortrag & Korrespondierender Autor:

Iris Hochstatter  
Universität der Bundeswehr München  
Institut für Technische Informatik  
Werner-Heisenberg-Weg 39  
85577 Neubiberg  
Tel.: 089-6004-3445  
Email: iris.hochstatter@unibw.de

# MetaVoIP – Sharing Contact Information over Organizational Boundaries

Frank Eyermann, Iris Hochstatter

Institut für Technische Informatik  
Universität der Bundeswehr München  
Werner-Heisenberg-Weg 39  
85577 München  
Frank.Eyermann@unibw.de  
Iris.Hochstatter@unibw.de

**Abstract:** Nowadays, many projects are carried out in a collaboration of people or groups from different institutions and/or enterprises. Such a *virtual organization* is characterized by high communication needs but does not operate in one single environment. MetaVoIP eases the communication as it automatically combines contact information from different PBXs and provides it to all partners. For a given virtual organization the MetaVoIP server is periodically retrieving contact information from the participating organizations via different data source drivers. The PBX connectors provide the telephony functionality independent of an organization's PBX, and the user GUI allows for easy user management as well as click-to-call. MetaVoIP has been implemented and tested; drivers exist for the Asterisk PBX and data input from LDAP and Asterisk configuration files.

## 1 Introduction

A cooperation of different enterprises, institutions or generally speaking organizations always increases the communication needs between those organizations. Nowadays, communication is already greatly supported by electronic media, such as email, wikis or special collaboration tools. Yet, voice communication is still the most important means of communication. Making a phone call, however, requires knowing the phone number of the organization and the extension of the callee. Within an organization a corporate directory typically satisfies this information need. Because of privacy and security issues these directories are mostly not publicly readable, leaving project partners calling a receptionist and being connected. Some projects might run project-specific directories, but those are usually not linked with the corporate one and therefore have to be updated manually whenever a change occurs. MetaVoIP closes this gap. The name MetaVoIP is modeled after the term metadirectory being a directory service, which combines the data from various directories and displays them in a uniform way [Je07]. MetaVoIP combines data from different private branch exchanges (PBXs) and provides a centralized phone book with additional user services. The central phone book is automatically built on the configurations of the local phone systems. The phone systems can be Voice-over-IP-based, but this is not a requirement.

The remainder of this paper is organized as follows: Section 2 discusses related work for the MetaVoIP functionality. We then follow up with the design of MetaVoIP, which includes the requirements, MetaVoIP topology and architecture. The section closes with in-depth details about the different abstraction layers and roles important within MetaVoIP. Section 4 gives implementation details about all components of MetaVoIP including security issues. We conclude the paper in Section 5 and give an outlook to future work.

## **2 Related Work**

MetaVoIP combines contact information for virtual organizations and keeps them up-to-date automatically. To our knowledge, no other approach is a solution to this particular problem. Still, the research areas of (meta-)directories and approaches to phone book applications are highly related and we have investigated them closely.

### **2.1 Directories and Metadirectories**

A directory service is a network service providing information about objects, e.g. a directory service implementing an address book provides information about people. The Lightweight Directory Access Protocol (LDAP) [LD97] allows querying and modifying directory services and organizes the data in a tree. LDAP is the replacement for X.500 and was specifically designed to be less complex and extensive. The data-model is object-oriented and the directory tree can be distributed over several servers.

A metadirectory service collects and integrates data from different directory services or databases. MetaVoIP is designed to manage contact information from various PBXs where its organizations form a kind of virtual organization. Examples include research institutions collaborating in a joint project or companies and their subcontractors. Metadirectories as part of identity management solutions exist for a variety of applications, both commercial [Ev08] [Mi08] [Ra08] [Or08] and open-source. MetaComm [Fr00] was an approach by Bell Labs researchers towards a metadirectory for telecommunications in 2000. It combined the metadirectory with the functionality to modify the data in the metadirectory and ensured data integrity at the same time. DS4J also offers metadirectory functionality and is implemented in JBoss [DS08]. [Ma03] describes a project that is related to the OpenLDAP community and presents a metadirectory gathering the data on a regular basis from heterogeneous sources.

### **2.2 Asterisk**

Asterisk [Va05] is an open-source PBX including all common telephony functionalities. Originally developed by Digium Inc., today many other developers have joined the project. Asterisk supports plain old telephone service (POTS) and Integrated Services Digital Network (ISDN) as well as VoIP with different protocols. Those features allow for various enhancements and easy extensibility and make Asterisk a very powerful tool.

LDAP support was introduced to Asterisk only in the most current version 1.6. Previous versions, which are still widely deployed, do not support LDAP [Vo08].

A great number of extensions to asterisk destined to administrators as well as users have been developed. An overview with more than 100 entries showing ready-to-use distributions, configuration managers, solutions for service providers, billing and call reporting, call center and contact center management solutions, status viewers and user interfaces can be found in [AG08]. However, none of these tools and extensions addresses the management of a federated environment consisting of several independent organizations and thus independent PBX and directory services.

### **3 Design of MetaVoIP**

Before we describe MetaVoIP in detail, we first describe a typical scenario and derive from this the requirements to a metadirectory approach supporting the transparent sharing of contact information over organizations. Second, the topology of a MetaVoIP virtual organization and architecture of MetaVoIP is described. The MetaVoIP authentication and authorization mechanisms, as well as its roles are given in the third subsection. At the end, the MetaVoIP data model is specified.

#### **3.1 Scenario and Requirements**

Within an enterprise or cooperation several subsidiaries or institutions (termed organizations in this paper) are collaborating. A single directory of all participants of all organizations is missing, because of legal or organizational aspects, or because only one branch or department is involved in the cooperation. Yet frequent voice communication between the partners is necessary. Each of the organizations operates its own private branch exchange, which might be based on any telecommunication technology. The PBXs are produced by different manufacturers and administrated independently by each organization. However, each PBX is connected to the network and provides an interface in order to remotely control the PBX.

Although no central directory for all participants exists, each partner itself operates a local directory. The directories may be based on different technologies, e.g., LDAP, relational databases or simple text files and can be accessed over the network. The data in the directory contains at least the persons' telephone numbers. A local directory is termed data source in this paper.

In order to simplify contacting project partner by phone MetaVoIP must show a list of all published users of all organizations. Organizations can limit the number of users published by disallowing MetaVoIP to read parts of the directory. Beneficial for users is the possibility to call another user with one click. Especially in large cooperations users of MetaVoIP can be further assisted with search and filter functionalities. If possible the status of an extension (e.g., offline, online, or busy) should be displayed, too. Because of privacy issues access to MetaVoIP must be authenticated and authorized.

### 3.2 MetaVoIP Topology

MetaVoIP is designed as a centralized server application. It abstracts from organizations and describes the topology with sites. For simplicity it is assumed that each organization is located at one site. A site is defined as a location with exactly one data source and exactly one PBX. Mappings must be made, if this assumption does not hold. An example with three sites is shown in Figure 1. The sites with each a data source and a PBX are connected to the MetaVoIP server. Users can access the services of MetaVoIP over the network.

### 3.3 Architecture

The architecture of MetaVoIP consisting of four main building blocks is depicted in Figure 2. An extended three-tier approach is used. The *User GUI* provides a graphical user interface and thus implements the presentation layer. The *Application logic* implements the business logic of MetaVoIP: Firstly, it serves the *User GUI* and supplies it with data from a local database. It performs the user commands, e.g., initiating a call with the help of the PBX abstraction layer and a *PBX connector*. Secondly, it periodically replicates data from the organizations in its own local database. The persistence layer of a three-tier architecture consists in MetaVoIP of three parts: a local database, which stores all data necessary for operation of MetaVoIP, a *Data source connector*, and a *PBX connector*. We will describe the data model of MetaVoIP in Section 3.5, and the two connectors subsequently.

#### 3.3.1 Data source connector

The *data source connector* provides an interface to the user information stored remotely at each site and abstracts from the concrete representation of the data. A more detailed view of the data source connector is shown in Figure 3. Different *data source drivers* may be implemented in order to connect to different types of data sources, e.g., LDAP, X.500, SQL database, or a driver to retrieve the user data directly out of a PBX.

The *data source abstraction layer* provides a uniform interface towards the application logic and chooses the appropriate driver depending on the *OrgSite.orgid* field (see Section 3.5). As each site operates by definition only one data source one driver is

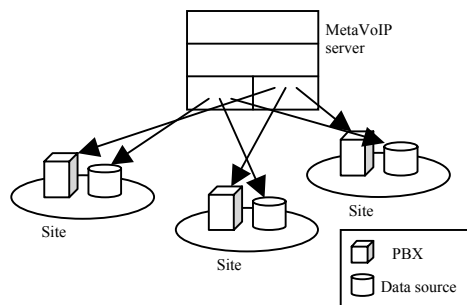


Figure 1. Topology example

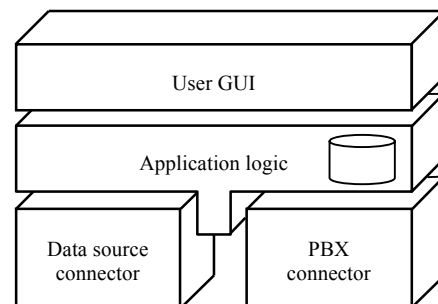


Figure 2. Architecture of MetaVoIP

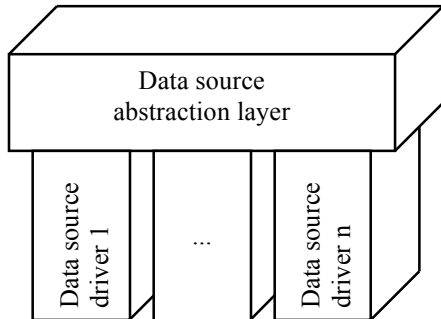


Figure 3. Data source connector

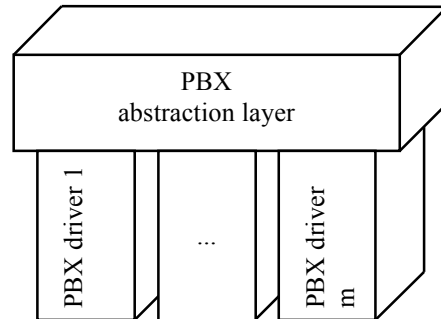


Figure 4. PBX connector

instantiated per site.

A data source driver must implement the function to read user information from the sites. User information consists of the list of all users, their attributes, as well as their phone accounts, i.e., the different telecommunication means with which the user can be reached. User passwords can only be replicated if they are stored as clear text in the data source. Only the data source itself can work with encrypted or hashed password. In this case if a user logs on to MetaVoIP the user is authenticated directly against the data source at his site (see also Section 4.4).

### 3.3.2 PBX connector

The *PBX connector* is the link between the application logic and the PBX system at a site. Because of the different types of PBX systems and the different access possibilities respectively, a PBX connector implements a driver concept, analogously to the one of the data source connector (see Figure 4).

A PBX driver implements two functions: First it periodically checks the status of a phone account, and second it can establish a phone call. The status of a phone account shows if a phone is currently reachable. This is of special interest for phones, which are not always connected as, e.g., SIP soft phones. In this case showing the connected state of phone accounts to a user will give him an impression, who is online and reachable and who is not. The known states are: up, down, busy, and unknown. The second function is directly derived from the requirement that users should be able to place a call by clicking the callee in the user GUI. The function instructs the PBX to establish a phone call between two or more named phone accounts.

### 3.4 Roles

MetaVoIP facilitates a role-based authorization scheme. The table below shows the roles and their access profile defined in the application. A user can own several roles at a time.

Role name	Description
User	This role is automatically assigned to each user. It allows logon to MetaVoIP and use the its standard features
UserSiteAdmin	A user with the role UserSiteAdmin can manage the accounts of all users in his site.
UserAdmin	Users with the role UserAdmin can manage all user accounts irrespectively of the site they belong to.
SiteAdmin	This role provides the ability to manage data source drivers and PBX drivers of the own site. The "own site" is the site the user account belongs to.
Admin	Admins are able to create, change and delete site definitions, as well as add, change and remove data source drivers and PBX drivers for all sites.

The Application logic checks the authorization for each action a user triggers. If possible, the user GUI hides those actions, which are not allowed for a user.

### 3.5 MetaVoIP Data Model

MetaVoIP stores a replication of all data read-in from the sites' data sources in a local database. This has several advantages but also disadvantages. On the one hand MetaVoIP can access local data much quicker than data stored remotely in the organizations. This shortens the time necessary to display a new screen drastically. Furthermore data replication can be performed in off-peak hours using resources when they are not needed otherwise. On the other hand data replication always bears the thread of inconsistent and outdated data. As the remote data sources do not cooperate in the replication process (e.g., by writing a transaction log) the replication algorithms get more complex. Nevertheless the replication approach was chosen, as it is still less complex and the time to display data would be too long otherwise. Then setting up MetaVoIP a compromise has to be made between too frequent updates taking too many resources and too few updates leading to too much outdated data.

The data model of MetaVoIP in UML is depicted in Figure 5. The class *User* stores the attributes of a user. *Firstname*, *lastname*, *email* and *comments* are text fields storing the respective information. The field *userid* stores a site-unique identifier for each user. *Secret* stores a password, which is used for authentication in case the data source can read the password from the data source. See section 4.4 for a more detailed description on how this field is used. The class *Role* stores the roles of the users. Because a user might own more then one role an n:m-relationship has to be modeled. The class *PhoneAccount* represents one possibility to call a user, i.e. an extension. A user might be reachable by different means, e.g., by public switched telephone network (PSTN), Session Initiation Protocol (SIP) or mobile phone. The field *protocol* stores a description, which of these output channels a PBX should use when calling the user. *Accountid* stores the address information, which is protocol dependent, e.g., a phone number for PSTN or a SIP-URL for SIP.

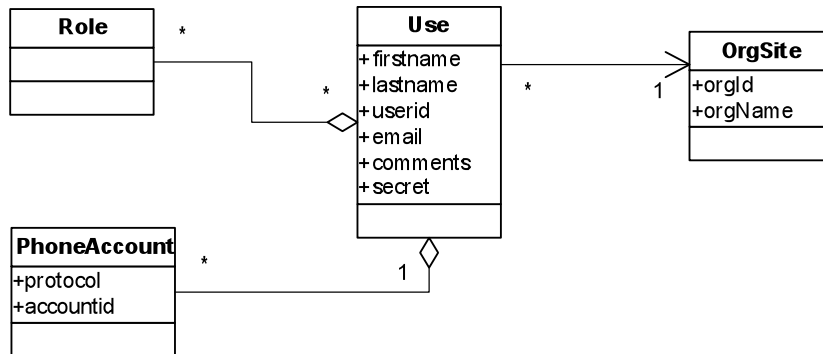


Figure 5. MetaVoIP data model

Each user is member of one site (class *OrgSite*). A site is identified by a unique identifier (*orgId*). The attribute *orgName* is only descriptive. The *orgId* in combination with *userid* from the class *User* is used to identify a user unambiguously.

## 4 Implementation

MetaVoIP is implemented as a web application using the programming language Java and an Apache Tomcat Server. Additionally, technologies from the Java Enterprise Edition are facilitated. Most prominent Java Server Pages (JSP) in combination with the Java Server Pages Standard Tag Library (JSTL) should be mentioned. The data model shown in section 3.5 is implemented using Java Beans. A bean each for *User*, *OrgSite* and *PhoneAccount* is defined. The roles are represented by a Java enumeration.

### 4.1 User GUI

The User GUI consists of web pages a user can access with his browser. This has the advantage that no additional software needs to be installed on client computers. The architecture described above is refined to implement the Model-View-Controller (MVC) concept using the framework Struts in version 1.3.8. A screen shot of the user GUI is shown in Figure 6. All screens are structured equally and they are divided into 5 parts: Part 1 displays a logo, part 2 offers links to log-in or log-off respectively. Part 3 is the navigation bar offering the user an easy way to navigate through the different functionalities of MetaVoIP. The navigation bar shows only those menu items the logged-in user is authorized for. Part 4 is the main part, displaying the actual content, e.g., the phone book. The information displayed depends on the menu item the user has chosen. The footer (part 5) displays status information. The User GUI features native language support. All text strings are separated from the source code and stored in a separate text file. Up to now translations to English and German already exist.

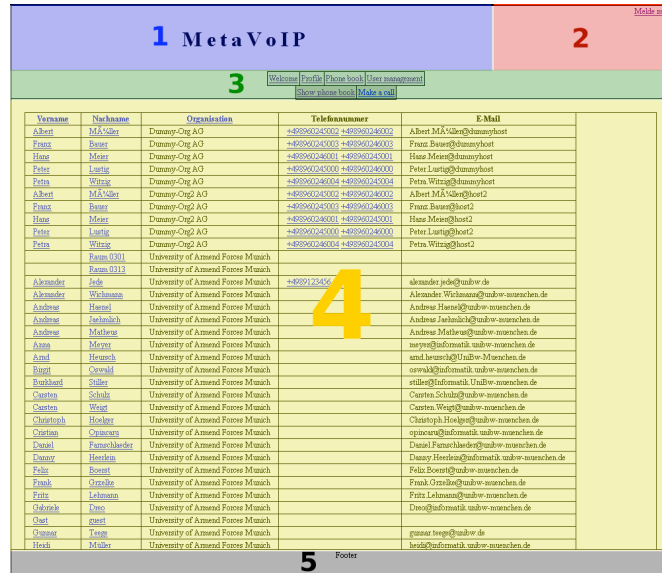


Figure 6. User GUI

#### 4.2 Implementation of the Data Source Driver

The data source abstraction layer is realized by a class called MetaVoipNet and a Java interface that all data source drivers need to implement. MetaVoipNet is responsible for choosing the appropriate driver for a site, loading it and supervising its operation. The abstraction layer therefore does not only abstract from the type of data source, it also abstracts from the number of data sources and drivers. Up to now two data source drivers are implemented: A driver to read the user data from an LDAP server and a driver to read the data from Asterisk configuration files. The LDAP driver stores roles with objects of type “OrganizationalRole”. The names of the objects have to be the name of the role with the prefix “metavoip”. The object for the role “User” is therefore “cn=metavoipUser”. All owners of the role have to be listed in the attribute “roleOccupant”. Phone accounts are stored in LDAP using the LDAP schema of Astirectory [Ad08]. Astirectory is an extension to the Asterisk PBX of previous versions, which allows storing SIP and IAX accounts in an LDAP server instead of the respective configuration files.

The second driver retrieves its data directly from Asterisk configuration files. The driver periodically downloads the Users, SIP and IAX configuration files from an Asterisk server. The download is performed via Secure Copy (SCP) requiring an operational SSH daemon on the Asterisk server. MetaVoIP then parses these files and extracts the relevant data. The roles a user occupies are stored in the users.conf file as comments of a defined format.

### **4.3 Implementation of the PBX Driver**

The PBX abstraction layer is realized by an abstract Java class, which already implements some common functionality. Drivers for a concrete PBX need to override this class and implement the abstract methods for establishing a call between two named phones and for getting the status of a phone account. Up to now a PBX driver for the Asterisk PBX is implemented. This driver uses the Asterisk Manager API for sending commands to the Asterisk PBX. The implementation uses the Asterisk-Java framework [Aj08], a Java wrapper library for the text based Asterisk Manager API. The Asterisk Manager API is quite unsecure because it operates only with clear-text passwords. Therefore a special proxy on the Asterisk server was implemented. Commands are transferred in a SSL-secured tunnel between the PBX driver on the MetaVoIP server and this proxy. The proxy decrypts the commands and forwards them to the Asterisk server.

### **4.4 Security**

The security requirements to MetaVoIP are moderate. MetaVoIP does not implement business critical processes and the organizations' PBXs can run without MetaVoIP. However, MetaVoIP stores user passwords and passwords to telecommunication equipment, which need to be protected. Furthermore unauthorized access to MetaVoIP could be misused. Two possible ways exist to authenticate users. If a data source stores clear text passwords, which could be retrieved by the data source driver, the passwords will be replicated to the MetaVoIP server. In this case the data source driver has to ensure that the transfer is protected, e.g., the data source driver for the Asterisk configuration files transfers them with Secure Copy (SCP).

In the other case if no clear text passwords could be retrieved by the data source driver (because of technical or policy reasons), the driver checks the password against the data source each time a user logs in. The LDAP driver uses this method. When a user tries to log on to MetaVoIP, the driver checks the password by authenticating against the LDAP server of the user's site. If this authentication succeeds, name and password is correct and the user is granted access to the application. The Asterisk PBX driver implements an additional proxy, which is installed on the asterisk server in order to protect the connection. See Section 3 above for a description of this proxy.

## **5 Conclusions**

MetaVoIP is a centralized server application, which can improve the integration between cooperating organizations. Its main focus is to loosely couple the PBX systems of the different organizations in order to make calls between organizations easier.

For this purpose, MetaVoIP uses a metadirectory approach that ensures short response time of the application. To avoid outdated information, data is automatically retrieved from data sources at the respective sites of the organizations. The organization keeps full control over its data and access rights.

MetaVoIP has been implemented as a prototype with two data source drivers (LDAP and Asterisk configuration files) and one PBX driver for Asterisk. The driver concept used for connecting PBX and data sources offers an easy way to implementing further drivers. Common functionality necessary for each PBX driver is already implemented in the base class. In addition, click-to-call functionality on a web site is also provided.

Next steps will be the implementation of additional drivers for data sources and PBXs. Also the migration of the LDAP data source driver to Asterisk release 1.6, which already includes LDAP support, will be a future task.

## Acknowledgments

This research activity has been performed partially in the framework of the EU IST Network of Excellence EMANICS “Management of Internet Technologies and Complex Services” (IST-NoE-026854).

## References

- [Ad08] Asterisk e.V.: Astirectory. Available: [http://www.asterisk-ev.org/projekte\\_astirectory.html](http://www.asterisk-ev.org/projekte_astirectory.html)
- [AG08] voip-info.org: Asterisk GUI. Available: <http://www.voip-info.org/wiki-Asterisk+GUI>
- [Aj08] Asterisk-Java. Available: <http://asterisk-java.org/>
- [As08] Asterisk Flash Operator Panel. Available: <http://www.asternic.org>
- [DS08] DS4J. Available: <http://sourceforge.net/projects/ds4j/>
- [Ev08] Evidian Identity and Access Management. Available: <http://www.evidian.com/iam/index.htm>
- [Fr00] Freire, J.; Lieuwen, D.; Ordille, J.; Garg, L.; Holder, M.; Urroz, H.; Michael, G.; Orbach, J.; Tucker, L.; Ye, Q.; Arlein, R.: MetaComm: a Meta-Directory for Telecommunications. In: Proc. 16th International Conference on Data Engineering, San Diego, California, USA, 2000, pp. 211-219.
- [JE07] Jede, A.: Design and Implementation of a framework to integrate corporate Asterisk systems. Master thesis, Information Systems Laboratory, University of Federal Armed Forces Munich, 2007.
- [LD97] Wahl, M.; Howes, T.; Kille, S.: Lightweight Directory Access Protocol (v3). Internet Engineering Task Force (IETF) RFC 2251, December 1997.
- [Ma03] Masarati, P.: OpenLDAP & Meta-directory. Presented at the OpenLDAP Developers' Day, Vienna, Austria, July 2003. Available: <http://www.openldap.org/conf/odd-wien-2003/ando.pdf>
- [Mi08] Microsoft Identity Lifecycle Manager 2007. Available: <http://www.microsoft.com/windowsserver/ilm2007/default.aspx>
- [Or08] Oracle Virtual Directory. Available: [http://www.oracle.com/technology/products/id\\_mgmt/ovds/index.html](http://www.oracle.com/technology/products/id_mgmt/ovds/index.html)
- [Pe08] Penrose Project. Available: <http://penrose.safehaus.org>
- [Ra08] RadiantOne Virtual Directory Server. Available: <http://www.radiantlogic.com/>
- [Va05] VanMegelen, J.; Smith, J.; Madsen, L.: Asterisk. The Future of Telephony. O'Reilly, 2005.
- [Vo08] voip-info.org: LDAP. Available: <http://www.voip-info.org/wiki/view/LDAP>