

# **AMETAS**

*Good Migrations.*

## ***AMETAS White Paper Series***

by Klaus Herrmann  
and Michael Zapf

# **On Places**

July 2000

Johann Wolfgang Goethe-Universität Frankfurt/Main, Germany  
Department of Computer Science

[www.ametas.de](http://www.ametas.de)

## Finding the Right Place to Work

You might have already read about AMETAS agents wandering (or using the technical term: *migrating*) from one host to another. But how do they specify their destination? Of course, you could say, they just use the host name. But what can you do if there are several places running on that host? And are you sure that the host name is always the same?

### Tasks of a Place

In order to run agents, some infrastructural component is necessary that compensates for the lack of code mobility by the operating system. These entities are called *places* in AMETAS. Places

- care for the message transfer between the Place Users;
- provide threads of execution for each running Place User;
- accept incoming agents and send agents to other places;
- communicate with user interfaces.
- utilize the security subsystem to prevent security breaches.

A place is a requirement for a host to participate in the agent system. This does not mean that all hosts without a place running on them are invisible to agent-based applications. For example, services can create socket connections and use them to connect to other facilities on remote hosts. This allows to collect information coming from hosts outside the agent system.

### Types of Places

All places have the same basic functionality as described above. Services may serve to extend the capabilities of a place. Due to restrictions of the underlying operating systems, places cannot support user adapters from more than one user that utilize a graphical display<sup>1</sup>. Therefore, a user adapter may normally only be used on the display of the host where the place is running.

In AMETAS, places may be used as *temporary places* or *permanent places*. While the former is intended for users to run their individual agents, services and user adapters, the latter should be seen as a part of a permanent agent system infrastructure. You can compare this differentiation with hosts on the Internet where you find hosts that are continuously reachable and those that connect to the Internet via a dial-up connection. Apart from their reachability, temporary and permanent places are functionally equivalent. As a consequence of the problem with graphical displays described above, user adapters are normally used on temporary places only.

A user may start his (or her) own temporary place e.g. on his laptop, start a user adapter and some agents and send the agents via a dial-up connection to another place of the agent system. Then the communication link and the temporary place may be

---

<sup>1</sup>In Windows, the graphics display always shows up on the screen that belongs to the host where the place is running. In Unix, you may redirect the output, but only to *one* target display per process. Places and all their Place Users are executed by only one Java Virtual Machine so that the situation is pretty much the same as in Windows.

shut down. When the agent has done its job, it wanders to a special place to be defined by the user and waits until the user's temporary place is reachable again. It then returns and present its results.

This provides the possibility to do disconnected operations which become increasingly important with mobile computing on the rise. Being off-line while your agent do some work for you may vastly cut connection costs.

## Relay Places

Temporary places may be temporarily unavailable, hence their name. Agents that want to migrate to an unknown or inactive place normally get an error message and the migration is canceled. A temporary place, however, can use another permanent place as a *relay place*, that is, it wants agents that intend to migrate to it to be sent to the relay place during its absence.

Relay places must be permanent. Permanent places do not have relay places. Agents may request to be reactivated at a relay place before some given time has passed. During this time, the relay place may attempt to forward the agent to the target place the agent meant to travel to<sup>2</sup>. When the agents are reactivated they should find out that they have not arrived where they intended to go and decide on some alternative actions.

## Place Name Service

Places cannot be directly named after the name of the host they are running on. This has mainly two reasons:

- One host should allow to run multiple places, probably from different users.
- The FQDN (Fully Qualified Domain Name) provided by the DNS (Internet Domain Name System) may not be constant at all times.

The second reason addresses a special problem of dial-up connections. Even if you are using the same provider to connect your host to the Internet, you'll normally get different IP addresses and also different FQDNs each time you re-connect. Agents that rely on this information would not be able to reliably return to your place (Figure 1).

AMETAS uses its own naming system called PNS (for *Place Name Service*) that maps place names to DNS host names and port numbers. This enables several users to run their place on the same host. When agents want to migrate, they have to use the place name registered in the PNS.

To deal with the second problem, the PNS is much more flexible than the Domain Name System which maps FQDNs to IP addresses. When a temporary place is started, it connects to a PNS server and transmits the currently valid location of the host and the port number. This way the place name can be preserved, but the mapping may be changed. The agent intending to return to a place does not realize that it is routed to a host address different to the one it departed from. To prevent attackers to have agents re-routed to their places the location update requires a secret key that was sent to the place at the previous time of connection.

---

<sup>2</sup>In the current implementation, agents are immediately reactivated at the relay place.

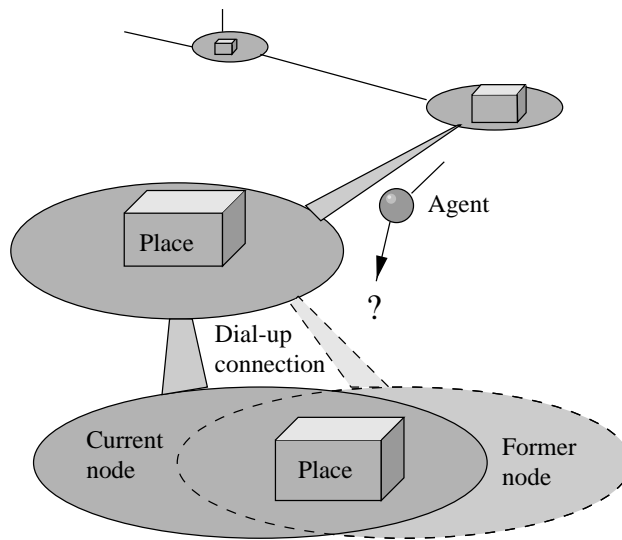


Figure 1: Same place, different host address

## Name Structure

Place names consist of two parts:

- the actual name of the place (also known as *simple place name*)
- the name of the domain containing this place (the *domain name*)

The simple place name and the domain name form the so-called *fully qualified place name*. Place names are assumed to be fully qualified by default. The domain name may consist of several subparts that are concatenated by dots. The place name and the domain name are also concatenated by a dot. The domain name ends with a dot.

Example: *myplace.subdom.dom.de.*

This denotes a place called *myplace*, being part of the domain *subdom.dom.de*. If the final dot is omitted, it is added automatically. The *empty* domain is therefore equivalent to the *root* domain “.”. Without domain specification, the place name *myplace* is assumed to be member of the root domain. As you probably noticed, the structure of place names is similar to the DNS name structure. However, keep in mind that the PNS names need not be related to any existing name in DNS.

When an agent wants to migrate to the above mentioned place, it passes that place’s name as an argument to the migration command. The agent does not have information about the actual location of the place in the network.

## Structure of the PNS

The Place Name Service consists of a class to be used by components of the agent system (*AMETASPlaceNameService*), standalone programs (*PlaceNameServer*), and an administrative tool (*PNSAdmin*). Place name servers must be running before any place is started. Each server uses a file that contains the domains it is responsible for

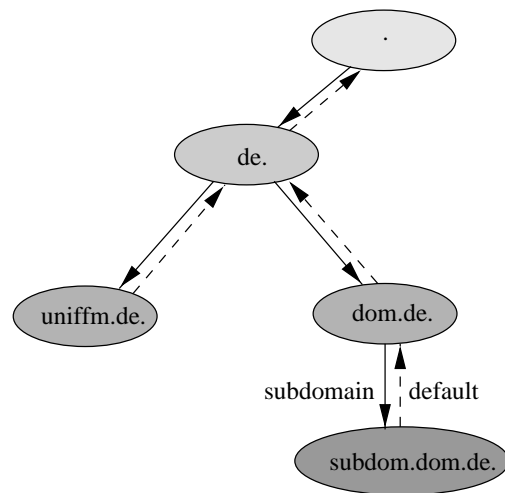


Figure 2: PNS structure

as well as a file for each of these domains, enumerating all places belonging to this domain. Clients may connect to *any* Place Name Server in order to submit requests.

When a client issues a request to a server, it checks at first whether it can answer the request by itself. If not, it checks if it knows a server that serves requests about a superdomain of the domain that the place belongs to (in the above example, *dom.de* is a superdomain). That failing, it forwards the request to a so-called *default server*. If the place could not be found, an error is returned to the requestor (Figure 2).

Servers manage a separate list for temporary places in a domain they are responsible for. They insert, modify, and delete entries accordingly. These lists may be queried any modified by the *PNSAdmin* tool that is part of the AMETAS distribution.

## PNS Administration

Sometimes, additional administration can be necessary. For example, the entries of temporary places are usually preserved in the lists of the place name servers because there could be agents that want to return to this place. However, there might be situations where this entry should be deleted, especially when you do some tests.

For this purpose, *PNSAdmin* is a Java standalone program that allows you to administrate a PNS server. With *PNSAdmin* you even need not connect with the respective PNS server; it suffices to connect to any PNS server that is part of your place name system. You can

- query the server for a list of places in a given domain;
- determine the host and port number of a give place;
- add new entries;
- delete entries.

Of course, removing entries requires either the secret key which was sent to the respective place or the *master password* that belongs to this PNS server.