Messaging for a Dynamic World

Mobeon Internal

Approved: Per Berggren		No: 3/FS-CRH 109 581/1 Uen	
Copyright Mobeon AB All rights reserved	Author: Tommy Lundemo Title: FS – Profile Manager	Version: PA5 Date: 2008-03-18	1/19

FS – Profile Manager

Content

History

Version	Date	Adjustments
PA1	2007-08-28	First version of the document. Original document was found in MAS. (EMAHAGL)
PA2	2007-09-19	Updated with attribute value control (EMAHAGL)
PA3	2008-02-26	Minor updates for segmented CoS (QTOMMLU)
PA4	2008-03-07	Updates after internal review (QTOMMLU)
PA5	2008-03-18	TA after review (QTOMMLU)

1 Introduction

This document specifies the function of the Profile Manager component. The Profile Manager provides an interface that can be used by other components in order to retrieve for example COS settings or a subscriber Profile according to the system data model (Technical Report M3 Data Model, 6/0363-PRO049 Uen) to create or delete subscribers from the system.

The Profile Manager communicates with MUR using LDAP in order to retrieve subscriber data.

1.1 Glossary

1.1.1 Profile

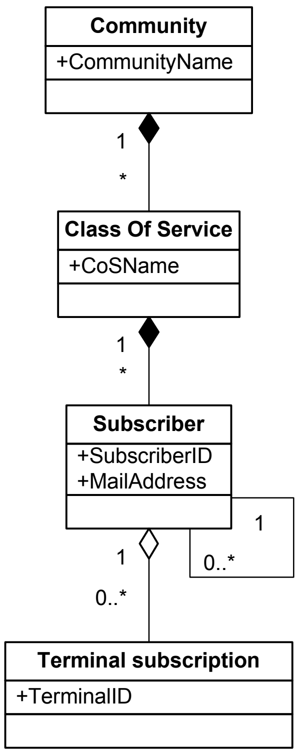

Figure 1 Profile

A profile is a grouping of all the settings applicable for a subscriber.

Given a unique identifier of a subscriber (i.e. SubscriberID or MailAddress) or a terminal subscription (i.e. TerminalID) a profile is built based on the settings in the Terminal Subscription, Subscriber, COS and Community for that subscriber.

According to [1], a subscriber (and therefore also a profile) has zero or more instances of a Mailbox as defined in [2].

2 Function Requirements (Commercial)

This paragraph is intentionally left blank.

3 Function Specification (Design Related)

3.1 Introduction

The Profile Manager exports the IProfile, ICOS, ICommunity and IDistributionList interfaces. These interfaces are used by clients to retrieve a subscriber Profile and the Community, COS or distribution lists of a Profile.

The Profile Manager imports the ILog, IConfig, IMediaObject, ILocateService, IServiceInstance, IMailbox, IEventDispatcher and IProvisioning interfaces.

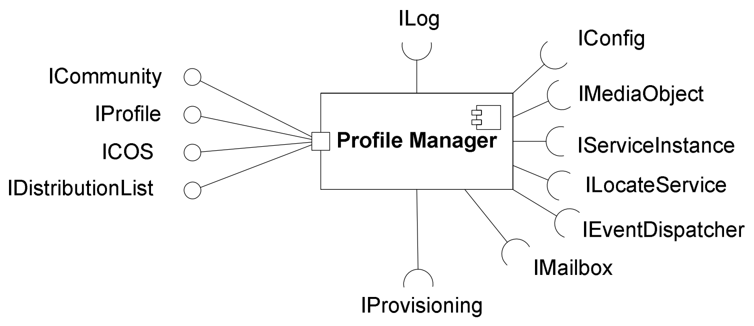

Figure 2 Profile Manager component

The client requests a Profile by specifying unique subscriber identification or terminal subscription. The Profile Manager returns the matching subscriber Profile to the client.

From the IProfile interface it is possible to retrieve separate parts of the profile; the Community or the COS. From the IProfile interface it is also possible to retrieve profile attributes such as distribution lists.

3.2 Exported Interfaces

The Profile Manager exports the following interfaces: IProfile, ICommunity, ICOS, and IDistributionList.

3.2.1 IProfile

The IProfile interface is used by a client to manage a subscriber Profile. IProfile offers the following methods:

- *IProfile[] getProfile(String entryPoint, AttributeFilter filter, Boolean limit)*
  A Profile instance is retrieved using a factory. The factory returns a list of the profiles that matches the given filter. The attributes allowed in the filter are configurable in the Profile Manager. The entryPoint is an optional parameter that indicates the LDAP search base used for the profile lookup. The limit parameter indicates if the search could be limited to the user directory in the local sub/domain, e.g. when using iMux (see section 3.4.4.1).

- *IProfile getProfile(String dn)*
  A Profile instance is retrieved for the passed dn.
- *Object getAttribute(String attrName)*
  Returns the requested attribute. Available attributes are described in section 8.
- *Set<Entry<String, String[]>>getAttributes()*
  Gets a the (aggregated) set of Attributes for this profile.
- *Set<Entry<String, String[]>> getAttributes(ProfileLevel profileLevel)*
  Gets only the set of Attributes for specified level (e.g. USER, COMMUNITY) in this profile. The aggregate of four consecutive calls to this function (with USER, BILLINGNUMBER, COS, COMMUNITY) is the same attributes as returned by method "getAttributes()" above.
- *void setAttribute(String attrName, Object attrValue)*
  Sets the specified attribute to the given value. Available attributes are described in section 8.
- *void createSubscription(ISubscription sub, String adminUID, String COSName);*
  Creates a subscriber with the specified characteristics. COSName is optional. In case the request fails, the client will be notified including the reason why.
- *void deleteSubscription(ISubscription sub, String adminUID);*
  Deletes a subscriber with the specified characteristics. In case the request fails, the client will be notified including the reason why.
- *ICommunity getCommunity()*
  Returns the community of the profile (see 3.2.2).
- *ICOS getCOS()*
  Returns the COS of the profile (see 3.2.3).
- *IMediaObject getGreeting(GreetingFilter filter)*
  Returns the greeting that matches the given filter. The properties allowed in the filter are specified in 2. IMediaObject is an imported interface, see chapter 3.3.
- *void setGreeting(GreetingFilter properties, IMediaObject greeting, IMediaContentResource resource)*
  Replaces the greeting that matches the given filter. The properties allowed in the filter are specified in 2. IMediaObject and IMediaContentResource are imported interfaces, see chapter 3.3.
- *IMediaObject getSpokenName(GreetingFilter filter)*
  Returns the spoken name of the profile. IMediaObject is an imported interface, see chapter 3.3.
- *void setSpokenName(GreetingFilter properties, IMediaObject spokenName, IMediaContentResource resource)*
  Sets the spoken name for the profile. IMediaObject and IMediaContentResource are imported interfaces, see chapter 3.3.
- *DistributionList[] getDistributionLists()*
  Returns the profiles distribution lists (see 3.2.4).

- *void addDistributionList(DistributionList distrList)*
  Adds a new distribution list (see 3.2.4).
- *void deleteDistributionList(DistributionList distrList)*
  Deletes the given distribution list.
- *IMailbox getMailbox(String mailHost, String accountID,*
  String accountPassword)
  Returns the requested Mailbox. The parameters mailHost and accountID uniquely identifies a Mailbox.
- *void close()*
  Indicates that resources allocated for the Profile can be freed: closing open mailboxes etc.

3.2.2 ICommunity

This interface is used by a client to get Community attributes. ICommunity offers the following methods:

- *ICommunity getCommunity(String community)*
  A community instance is retrieved using a factory.
- *Object getAttribute(String attrName)*
  Returns the requested attribute. Available attributes are described in section 8.

3.2.3 ICOS

This interface is used by a client to get COS attributes. ICOS offers the following methods:

- *ICos getCos(String cosDn)*
  A COS instance is retrieved using a factory. It is recommended to use getAttribute on profile instead, if the segmented COS model is used.
- *ICos getCos(String cosDn, String [] compoundServiceDn)*

  A COS instance is aggregated from the provided cosDn, including possible override settings defined by compoundServiceDn (applicable only to segmented COS).
- *Object getAttribute(String attrName)*
  Returns the requested attribute. Available attributes are described in section 8.
- *Set<Entry<String, String[]>> getAttributes()*
  Returns the set of attribute names and content interpreted as strings.

Note: If a segmented COS model is used, the ICos will hide the structure of the COS, always returning settings from the segmented COS model. Old and new model can coexist.

3.2.4 IDistributionList

This interface is used by a client to modify a distribution list. IDistributionList offers the following methods:

- *DistributionList(String ID)*
  Creates a distribution list with the specified identifier.
- *String getID()*
  Returns the identifier of the distribution list.
- void *addMember(String member)*
  Adds a member.
- *void removeMember(String member)*
  Removes the indicated member.
- *String[] getMembers()*
  Returns a list of the current members.
- *IMediaObject getSpokenName()*
  Returns the distribution list spoken name. IMediaObject is an imported interface, see chapter 3.3.
- *void setSpokenName(IMediaObject spokenName, IMediaContentResource resource)*
  Sets the distribution list spoken name. IMediaObject and IMediaContentResource are imported interfaces, see chapter 3.3.

3.3 Imported Interfaces

The Profile Manager uses the following external interfaces:

- ILog for logging
- IConfig for configuration
- IMediaObject for passing media objects such as a greeting
- ILocateService for M3 service lookup
- IServiceInstance for accessing service properties
- IMailbox for initiation of a subscriber Mailbox
- IEventDispatcher for retrieving events
- IProvisioning for create and delete of subscribers

3.4 Functions

3.4.1 LDAP Communication with MUR

The Profile Manager communicates with MUR using LDAP. An internal interface IDirectoryService is used towards the LDAP implementation.

The communication with MUR uses a configurable number of connections (using a connection pool) and does as few read and writes as possible (using data caching). The lifetime of a connection is configurable.

3.4.2 Referrals and Other LDAP Features

The Profile Manager LDAP communication support referrals.

The Profile Manager sets a connection timeout for the LDAP connections to avoid connections that will hang forever if MUR is down. The value of the timeout depends upon if the connection is used for reading or writing.

If an LDAP command fails, the Profile Manager can re-try the command. The decision whether or not to retry the command depends upon the time elapsed since the previous command was initiated and how many retries that have been configured.

3.4.3 MUR Host

The Profile Manager retrieves the MUR host to communicate with using the ILocateService interface. The Profile Manager asks the component register for a host and it is the component register that is responsible for determining which MUR hosts that is appropriate depending upon for example replicationid (for the MultiMaster function).

The Profile Manager asks the Component Register for a host each time to read from or write to MUR, see Figure 3.

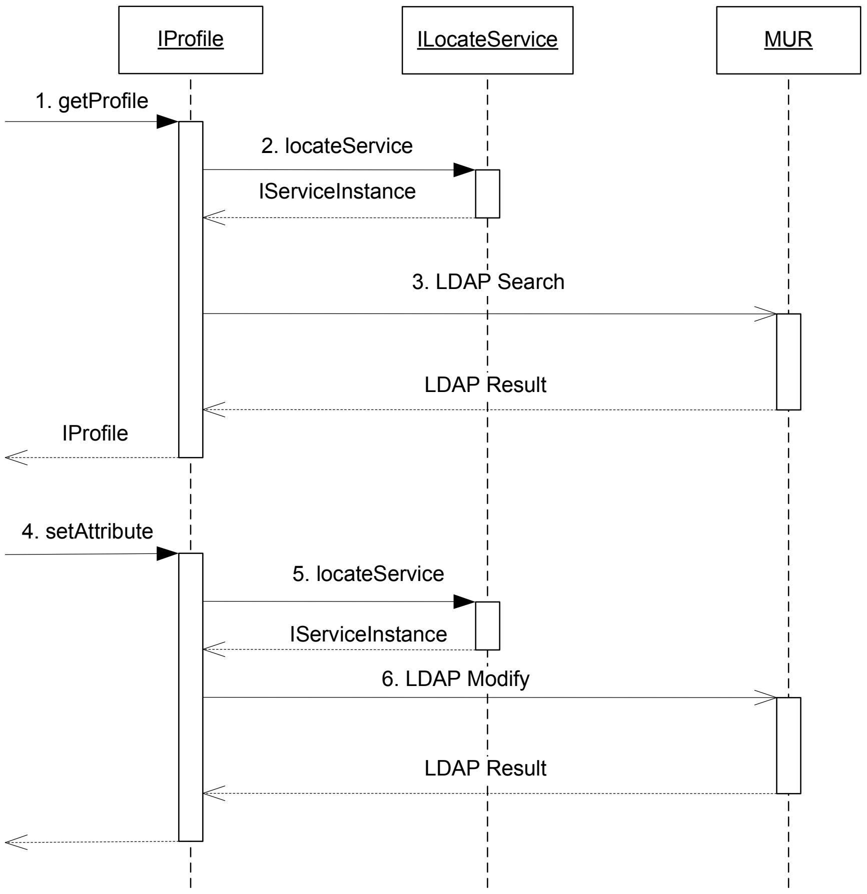

Figure 3 Host lookup in Component Register

1. A profile instance is requested.
2. The MUR host to use is retrieved using the ILocateService interface.
3. The LDAP search request is sent to the MUR host.
4. An attribute is changed in the profile.
5. The MUR host to use is retrieved using the ILocateService interface.
6. The LDAP modify request is sent to the MUR host.

3.4.3.1 MUR host errors

When communication with a MUR host fails, the Profile Manager reports this to the Component Register and retries the search request, see Figure 4.

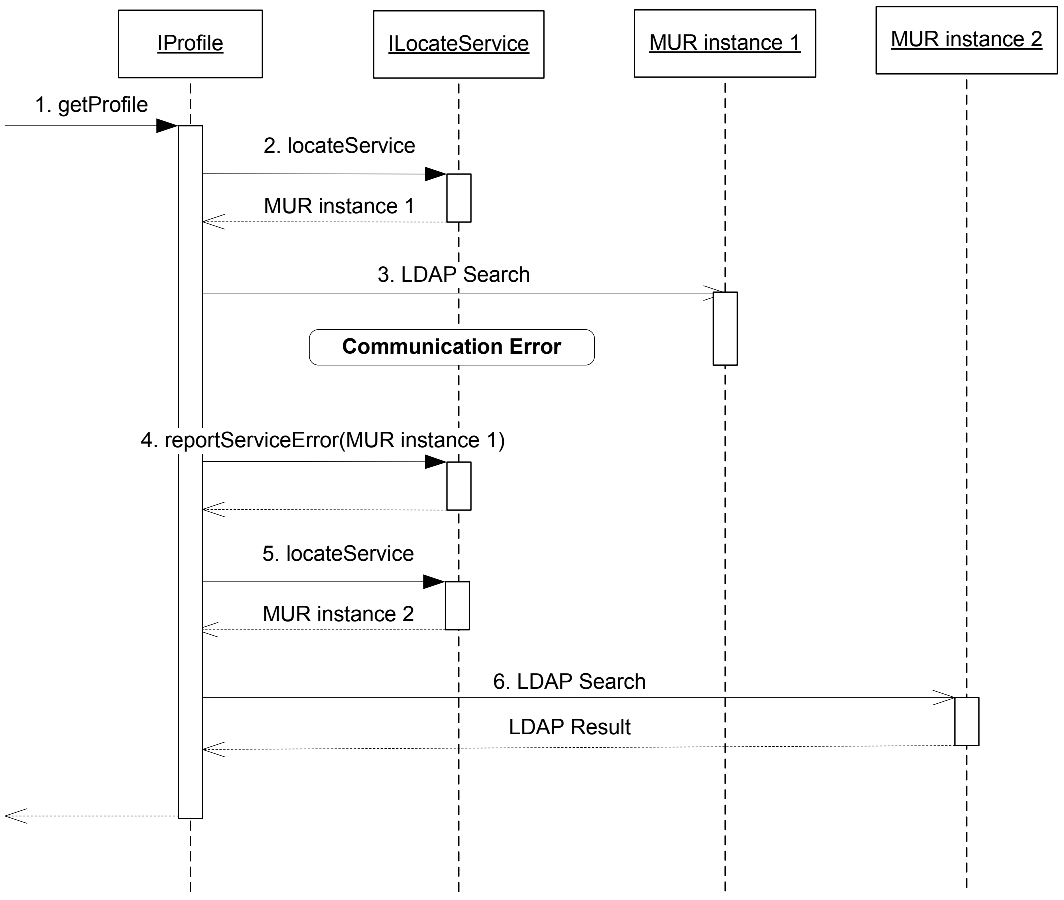

Figure 4 Host failure

1. A profile instance is requested.
2. The MUR host to use is retrieved using the ILocateService interface.
3. The LDAP search request sent to the MUR host fails.

4. The host failure is reported using the ILocateService interface.

5. Another MUR host is retrieved using the ILocateService interface.

6. The LDAP search request is sent to the MUR host.

If no MUR host can be retrieved using the ILocateService interface this is logged as an error and notified to the user of the Profile Manager.

3.4.4 Profile Lookup

The attribute or attributes to search for when finding a profile can be selected from the configured available MUR attributes, see 3.4.7.

A profile consists of the summarized data from Community, COS, Subscriber and Terminal Subscription. If an attribute exists on several levels, only one value is active for the profile.

When a profile is retrieved using method getProfile, the lookups of Community, COS, Subscriber and Terminal Subscription are made in a configurable priority order for each attribute.

When an attribute exists on several levels, a value with lower priority will be overwritten. For example, an attribute exists on both COS and Subscriber level. If the Subscriber level has higher priority, this attribute will be returned from the Profile. Of course the COS attribute can be retrieved using the ICOS interface.

If a profile cannot be created due to provisioning errors this is logged as an error and is notified to the user of the Profile Manager.

If a Segmented COS model is used the getProfile will internally get the correct profile and return it on the same format as for a non-segmented cos. Overrides on user level will be considered if COS is retrieved based on a user profile.

3.4.4.1 Support for iMux solution

A profile lookup can be marked as limited which means that the search is limited to a user-directory in a local subdomain. This is indicated by adding the LDAP attribute *limitScope* to the search. It is configurable if a limited profile lookup should use the *limitScope* attribute or not.

3.4.5 Subscriber creation and deletion

The Profile Manager provides functionality to create and delete subscribers from the system. The IProvisioning interface is used for this purpose. When a subscriber is to be created the Profile Manager basically performs the following tasks:

1. Adapt the ISubscription object (provided in the create request) using information specified in section 8 to the IProvisioning interface. Basically this means that the attribute name will be mapped to their corresponding names defined in the IProvisioning interface. The attribute value will be left unchanged and passed as is to the IProvisioning interface.
2. Looks-up the password to use for the provisioning administrator uid (provided in the create request). This information is then cached in the

Profile Manager and possible to use in consecutive sessions. The cache is updated at the same interval as other cached MUR data (e.g. COS).

3. If the client provided a COSName as well, Profile Manager, by using the community of the provision administrator provided, retrieves the COSDN from MUR. The COSDN will then be used in the IProvisioning interface.
   If the client provides both COSName and COSDN (as a Subscription parameter), the COSName will be used. If COSName is not found in the request will fail.
   If the client provides neither COSName nor COSDN, the administrators own COSDN (an attribute in the administrators profile that explicitly describes the COSDN to use when subscribers are to be created) will be used.

When a subscriber shall be deleted, only task 1 and 2 are performed.

3.4.6 Authentication

The Profile Manager does not do authentication of its clients. It is up to the clients (based on information provided by the Profile Manager) to authenticate end-users that shall have access to the subscriber information.

3.4.7 Available MUR Attributes

The MUR and provisioning attributes that shall be accessible to clients of the Profile Manager are listed in an XML document that is read at startup or when re-loading configuration. The XML document lists all MUR and provisioning attributes that should be available. For each attribute the document also lists the corresponding name in the Profile Manager, the type and format, the attribute read location (COS, Subscriber etc.), if this is a provisioning attribute, read location priority, write location, whether the attribute is read-only, max and min values and possible default values if such exists.

For String and Integer attributes a special attribute value control can be configured see 3.4.7.1

The main reason for keeping the attributes in an XML/configuration file is that it should be easy to extend for example the Profile Manager with support for a new MUR attribute.

3.4.7.1 Attribute value control

When setting a value to an attribute Profile Manager checks that the value is valid according to the syntax and range rules that are configured. If not configured all values are allowed.

Integer attributes have a configurable range check where min and max value can be set.

String attributes have a configurable syntax check. A regular expression can be set where the value must match this expression to be valid. Some attributes may require very complex syntax checks (for example checking a date value). So these checks are made a little simpler by just checking correct characters.

Note that Boolean attributes does not have any value control, they have their own configuration when setting values for true or false.

3.4.8 Greeting and Spoken Name

Greetings and Spoken Names follow the format described in [1].

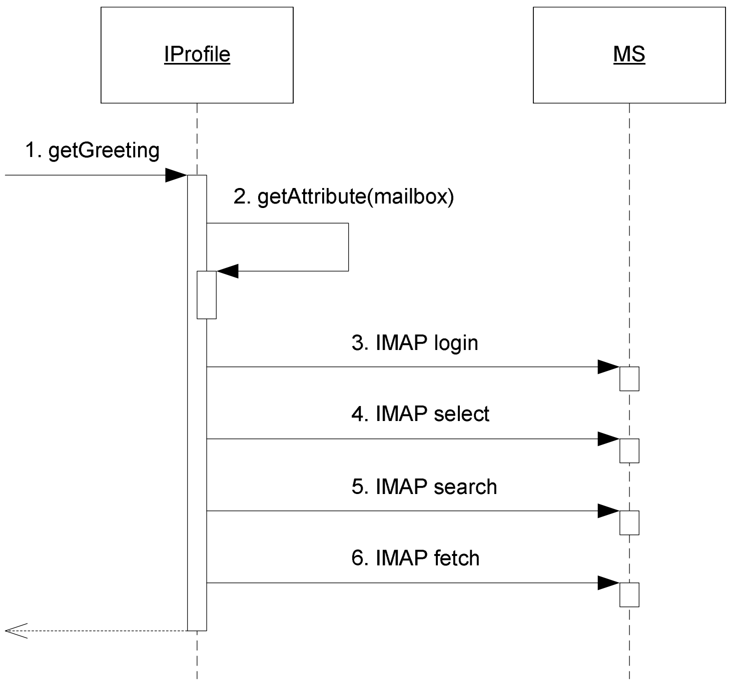

Figure 5 Retrieval of greeting

1. A greeting is requested (a spoken name request is handled in the same way).
2. The mailbox name containing the profile’s greetings is retrieved.
3. The mailbox is retrieved.
4. The greeting folder is selected.
5. The greeting message is searched in MS.
6. The greeting message is retrieved from MS.

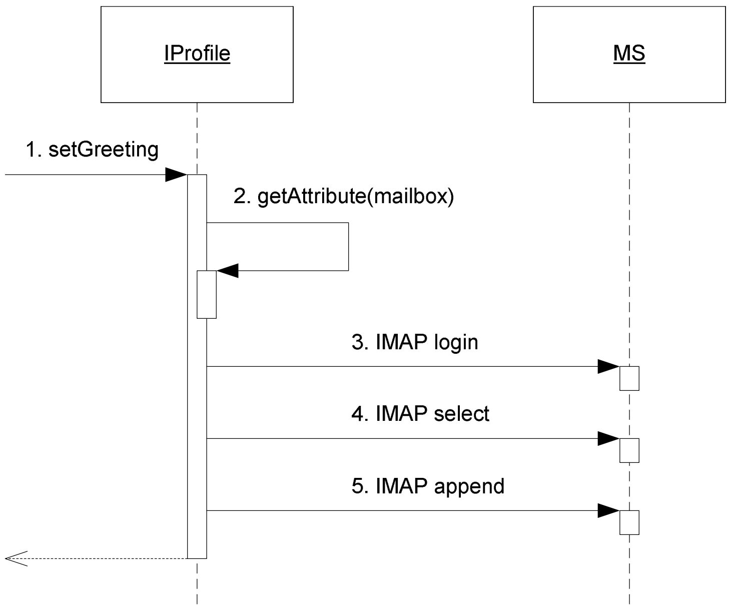

Figure 6 Setting a greeting

1. A greeting is submitted (a spoken name submission is handled in the same way).
2. The mailbox name containing the profile's greetings is retrieved.
3. The mailbox is retrieved.
4. The greeting folder is selected.
5. The greeting message is appended to the folder.

3.4.9 Available Greeting Properties

The client to the Profile Manager can access greetings that match a certain filter. Table 1 lists the properties used in the GreetingFilter.

Table 1 Greeting Properties

Property Name	Type	Valid values	Description
Type	String	"AllCalls", "NoAnswer", "Busy", "OutOfHours", "ExtendedAbsence", "CDG" or "Temporary"	Type of greeting.
Format	String	"voice" or "video"	The greeting format.
CallerId	String	A telephone number	Contains the number of the caller that shall receive a Caller Dependent Greeting. Only interesting to use in filter when accessing greeting of type CDG.

3.4.10 Distribution Lists

Distribution lists can be added, retrieved, removed and modified, see Figure 7 to Figure 10. Changes made in existing distribution lists are propagated to MUR.

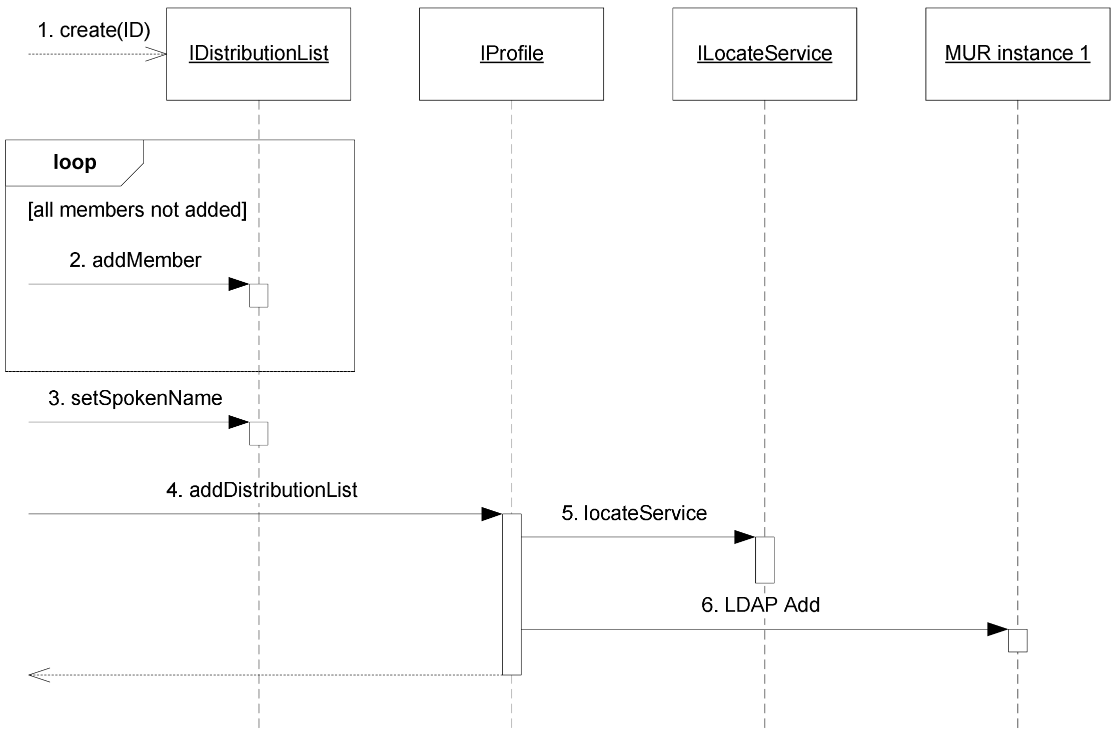

Figure 7 Adding a distribution list

1. A distribution list identified by ID is created.
2. All members are added to the distribution list.
3. The distribution list’s spoken name is added to the distribution list, see section 3.4.8 for how the spoken name is stored.
4. The distribution list is added to the Profile.
5. The MUR host to use is retrieved using the ILocateService interface.
6. The LDAP add request is sent to the MUR host.

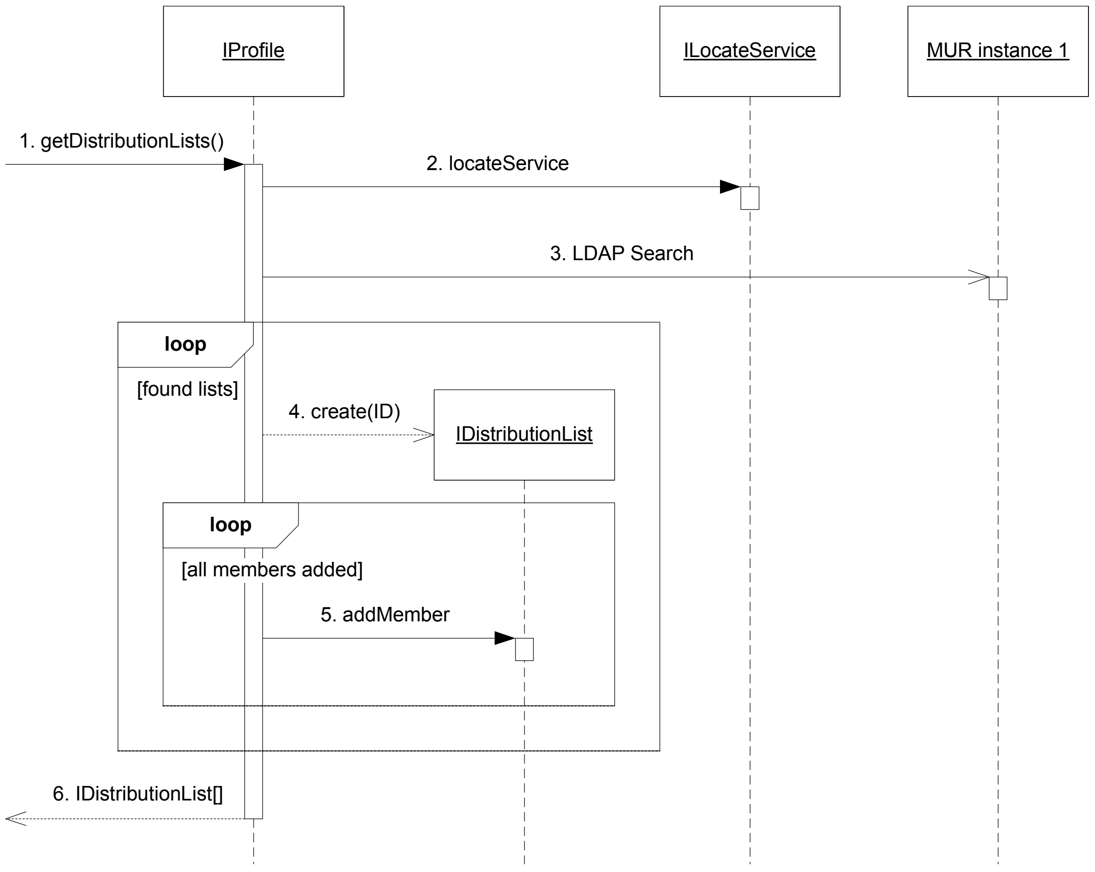

Figure 8 Retrieving distribution lists

1. The distribution lists are requested from the Profile
2. The MUR host to use is retrieved using the ILocateService interface.
3. The LDAP search request is sent to the MUR host.
4. A distribution list is created.
5. All members are added to the distribution list.
6. The distribution lists are returned.

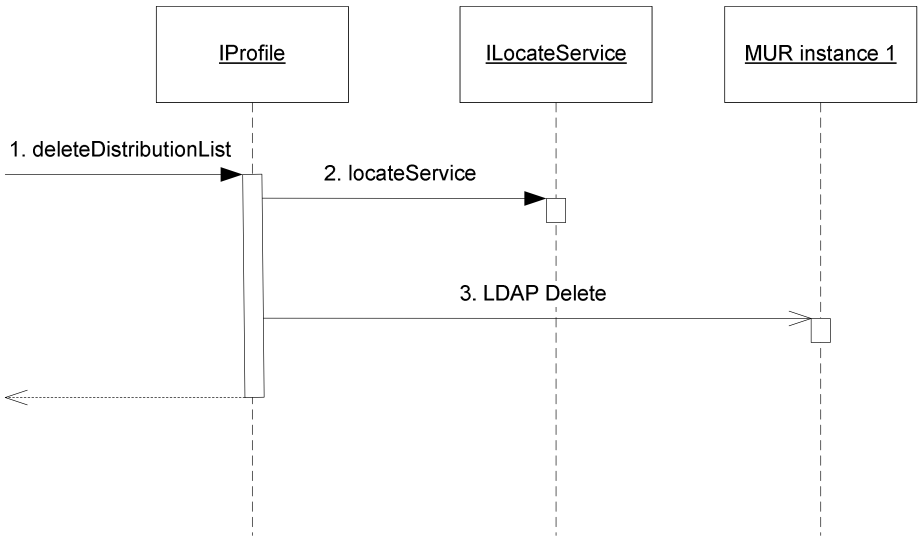

Figure 9 Deleting a distribution list

1. A distribution list is submitted to the Profile for removal.
2. The MUR host to use is retrieved using the ILocateService interface.
3. The LDAP delete request is sent to the MUR host.

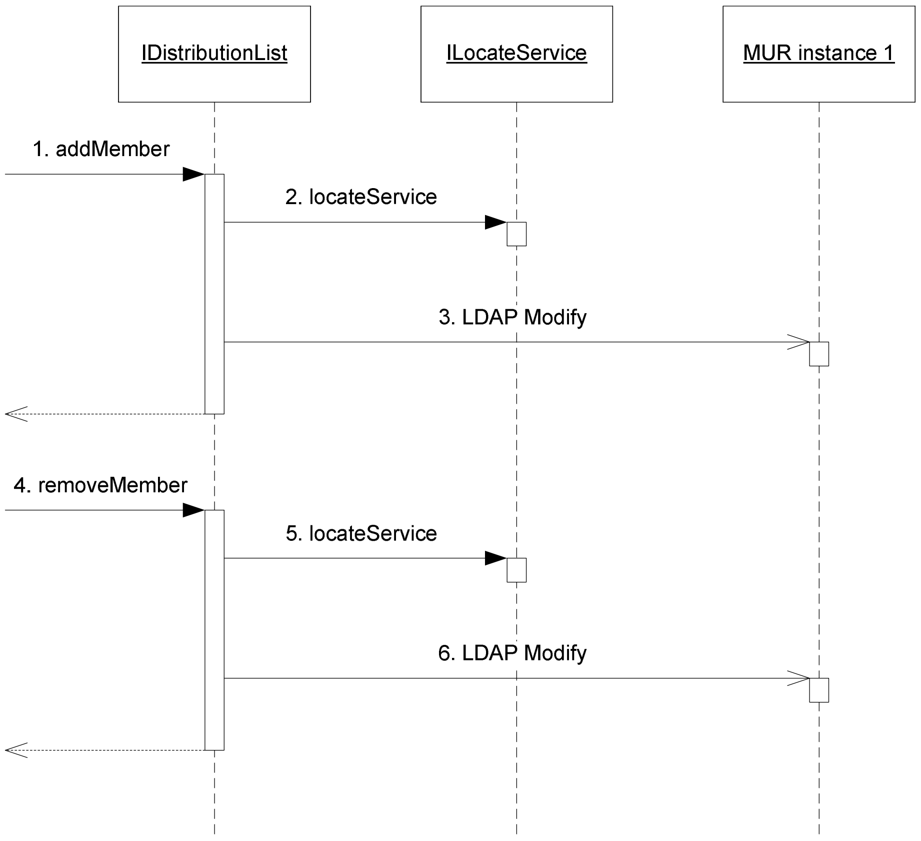

Figure 10 Modifying a distribution list

1. A new member is added to an already existing distribution list.
2. The MUR host to use is retrieved using the ILocateService interface.
3. The LDAP modify request is sent to the MUR host.
4. A member is removed from an already existing distribution list.
5. The MUR host to use is retrieved using the ILocateService interface.
6. The LDAP modify request is sent to the MUR host.

3.4.11 Events

3.4.11.1 Consumed

The Profile Manager registers itself using the IEventDispatcher interface as a receiver of the following events:

- Configuration has changed (indicates that configuration should be re-read)

3.4.11.2 Produced

- The Profile Manager does not produce any events.

3.4.12 Thread safe

The Profile Manager is thread safe.

4 External Operation Conditions

4.1 Configuration

The following are configurable in the Profile Manager:

- Maximum amount of simultaneous LDAP connection towards one MUR instance.
- Idle timeout for a connection, i.e. how long the connection can be unused before it is considered idle and is disconnected.
- Lifetime for a connection, i.e. how long the connection can be used before it is disconnected. This can be used to renew connections behind a load balancer for example.
- Read and write timeouts on a connection
- Number of retries in case of communication error
- The maximum time period during which retries are performed
- Directory administrator credentials
- The available attributes for a profile. These attributes are also possible to use for the search criteria for profile lookup.
- If the additional attribute *limitScope* should be added to profile searches marked as limited.

4.2 Logging

- Hosted service circumstances are logged as described in ref.[6] using the IHostedServiceLogger interface.
- All calls to interface methods are logged at info level. Method name and parameter values are logged. At return returned value is logged.

5 Capabilities

This paragraph is intentionally left blank.

6 References

[1] IWD End User Message Format
3/155 19-HDB 101 02 Uen

[2] FS Mailbox
4/FS-CRH 109 581/1 Uen

[3] FS External Component Register
6/FS-CRH 109 581/1 Uen

[4] MUR Developers Guide
3/1551-CRH 109 086 Uen

[5] FS Operate and Maintain Manager
17/FS-MAS0001 Uen

[6] FS-Log Manager
2/FS-CRH 109 581/1 Uen

[7] IWD-Provisioning Interface
2/155 19-CRH 109 086 Uen

7 Terminology

COS	Class of Service
LDAP	Lightweight Directory Access Protocol
MUR	Messaging User Registry
CAI	Customer Administration Interface

8 Appendix: Attribute Mapping

The mapping between attributes available in the IProfile interface and their corresponding names used in MUR and the CAI interface (see [7]) is described by the backend.xml and backend.xsd files. Each component using the backend can keep its own version of backend.xml and backend.xsd. The example below illustrates how to add a new boolean attribute foo by modifying these files.

First modify backend.xml:

```
<profilemanager limitscope="false" coscachetimeout="300000">
```

```
        <userregister readtimeout="5000" writetimeout="5000"
admin="cn=directory manager" password="emmanager"
defaultsearchbase="o=ericsson.com" trylimit="3" trytimelimit="500"/>
        <connectionpool maxsize="25" connectionlifetime="300000"/>
        <provisioning password=""/>
        <attributemap searchorder="community,cos,user,billing">
           <foo userregistername="foo" type="boolean" true="yes"
false="no" provisioningname="FOO"/>
        </attributemap>
    </profilemanager>
```

This means that the backend name for the attribute is foo, the MUR name is also foo and the provisioning name for our new parameter is FOO. To allow this attribute and to control if this attribute is writeable or not a row is needed in backend.xsd too:

```
<xs:element name="attributemap">
   <xs:complexType>
      <xs:all>
         <xs:element name="foo" type="booleanmetadatareadonly"/>
      </xs:all>
      <xs:attribute name="searchorder" type="searchorder" use="required"/>
      <xs:attribute name="basicattributes" type="xs:string"/>
   </xs:complexType>
</xs:element>
```

The attribute called "basicattributes" above is not necessary to specify in backend.xml. It has a default value that denotes which of the user profile attributes that are valid in a segmented COS model. (Currently: "passwordminlength,passwordmaxlength,maxloginlockout,inhoursstart,inhoursend,inhoursdow,diskspaceremainingwarninglevel,emexpirationrule,emretentiontime,emnoofmailquota,mailquota,emnotretrievedvoicemsg,cosname")