

# List Management and Do-not-Disturb V2.0.6 (2004-06)

---

*Technical Specification*

## **Push to Talk over Cellular (PoC); List Management and Do-not-Disturb; PoC Release 2.0**

---

Comneon, Ericsson, Motorola, Nokia and Siemens

---

Keywords

---

PoC, Group and List Management,  
Do-not-Disturb

---

**Copyright Notification**

No part may be reproduced except as authorized by written permission of the contributing companies.

© Comneon, Ericsson, Motorola, Nokia and Siemens.  
All rights reserved.

# Contents

Foreword .....	5
1 Introduction .....	6
2 Scope .....	6
3 References .....	6
4 Definitions and abbreviations .....	7
4.1 Definitions .....	7
4.2 Abbreviations .....	7
4.3 Requirement vocabulary .....	7
5 List management .....	7
5.1 Contact list management .....	7
5.1.1 Contact list attributes .....	8
5.1.2 Contact list identities .....	8
5.1.3 Contact list operations .....	9
5.1.4 Contact list policies .....	9
5.2 Access list management .....	9
5.2.1 Access list attributes .....	10
5.2.2 Access list identities .....	10
5.2.3 Access list operations .....	10
5.2.4 Access list policies .....	11
5.2.5 Processing of access lists .....	11
5.3 Group list management .....	13
5.3.1 Group attributes .....	14
5.3.2 Group identities .....	15
5.3.3 Group operations .....	15
5.3.4 Group policies .....	15
6 Do-not-Disturb .....	16
6.1 Do-not-Disturb attribute .....	16
6.2 Do-not-Disturb identity .....	16
6.3 Do-not-Disturb operations .....	16
6.4 Do-not-Disturb policies .....	16
6.5 Processing of Do-not-Disturb flag .....	16
7 Description of the list management protocol .....	17
7.1 General protocol requirements .....	17
7.2 Authentication .....	17
7.3 Protocol definition .....	18
7.3.1 Attribute manipulation .....	19
7.3.1.1 Set attributes .....	19
7.3.1.2 Get attributes .....	20
7.3.2 Contact lists .....	21
7.3.2.1 Create Contact list .....	21
7.3.2.2 Delete Contact List .....	22
7.3.2.3 Add user/group identity to contact list .....	22
7.3.2.4 Delete user/group identity from contact list .....	23
7.3.2.5 Retrieve Contact List .....	24
7.3.2.6 Modify user/group attributes in contact list .....	25
7.3.2.7 Get list of contact lists .....	25
7.3.2.8 Modify contact list attributes .....	26
7.3.3 Access Lists .....	27
7.3.3.1 Add user/group identity to user accept list .....	27
7.3.3.2 Delete user/group identity from user accept list .....	28
7.3.3.3 Modify user/group attributes in user accept list .....	28
7.3.3.4 Retrieve user accept list .....	29
7.3.3.5 Add user/group identity to user reject list .....	30

7.3.3.6	Delete user/group identity from user reject list.....	30
7.3.3.7	Modify user/group attributes in user reject list.....	31
7.3.3.8	Retrieve user reject list.....	31
7.3.3.9	Modify Access List Attributes.....	32
7.3.4	Groups.....	33
7.3.4.1	Create Group.....	33
7.3.4.2	Delete Group.....	35
7.3.4.3	Add user identity to group member list.....	35
7.3.4.4	Delete user identity from group member list.....	36
7.3.4.5	Modify user attributes in group member list.....	37
7.3.4.6	Retrieve group member list.....	37
7.3.4.7	Add user list identity to group reject list.....	38
7.3.4.8	Delete user identity from group reject list.....	39
7.3.4.9	Modify user attributes in group reject list.....	39
7.3.4.10	Retrieve group reject list.....	40
7.3.4.11	Get list of groups.....	41
7.3.4.12	Modify group attributes.....	41
7.3.5	Caching support.....	43
7.4	Error cases.....	44
7.4.1	Conflicts.....	44
7.4.2	Status codes and descriptions.....	45
7.5	Protocol XML schemas.....	45
7.5.1	Content application/buddylist+xml.....	45
7.5.2	Content application/buddylists+xml.....	46
7.5.3	Content application/buddies+xml.....	46
7.5.4	Content application/access-lists+xml.....	47
7.5.5	Content application/group+xml.....	47
7.5.6	Content application/groups+xml.....	48
7.5.7	Content application/attributes+xml.....	48
7.6	Protocol transport bindings.....	49
7.7	Typical UE to GLM procedures.....	49
7.7.1	Contact list management.....	49
7.7.1.1	Creating new contact list.....	49
7.7.1.2	Reading contact lists.....	49
7.7.1.3	Modifying an existing contact list.....	50
7.7.2	Access list management.....	51
7.7.2.1	Reading access lists.....	51
7.7.2.2	Modifying access lists.....	51
7.7.3	Group list management.....	52
7.7.3.1	Creating new group.....	52
7.7.3.2	Reading groups.....	52
7.7.3.3	Modifying group lists.....	53
7.7.4	Caching mechanism.....	53
7.7.4.1	Verifying locally stored data.....	53
7.7.4.2	Modifying data.....	54
7.7.5	Start-up procedure.....	54
8	Im – Is correlation issues.....	55
8.1	Server addresses.....	55
8.2	User identities.....	55
8.3	Contact list identities.....	55
8.4	Group identities.....	55
8.5	Authentication schemes.....	56
<b>Annex A (normative): UE configuration parameters for list management .....</b>		<b>56</b>
<b>Annex B (informative): Change history .....</b>		<b>57</b>

---

## Foreword

This Technical Specification has been produced by Comneon, Ericsson, Motorola, Nokia and Siemens.

---

# 1 Introduction

This specification contains the procedures, signaling flows and signaling parameters for the Group and List Management Server, a part of the Push to Talk over Cellular (PoC) service, and List Management protocol on the Im interface defined in the reference [2].

The document is divided into the following parts:

Part 1 (Clause 5): “List management” part where the elements managed on the Im interface and the operations performed on these elements over Im interfaces are described.

Part 2 (Clause 6): “Do-not-Disturb” part where the Do-not-Disturb flag and the operations performed on it over the Im interface are described.

Part 3 (Clause 7): “Description of the list management protocol” part where the protocol used on the Im interface is defined.

Part 4 (Clause 8): “Im – Is correlation issues” part where correlation between items and addressing on the Im and Is interface is defined.

Part 5 (Annex A): “UE configuration parameters for list management”. This part describes a minimum set of parameters required by this specification to be configurable in the UE as a prerequisite for the procedures in parts 1-4.

---

## 2 Scope

List management and Do-not-Disturb has functions that allow the end user or the operator to affect the behavior of the PoC features (see [1]). Certain features’ behavior is affected, controlled, or realized using lists and attributes (e.g. Do-not-Disturb) that contain information manipulated by the user or the operator. Management of these lists and attributes over the Im (UE – GLMS, see [2]) interface is the subject of this specification.

This specification is part of PoC Release 2.0.

---

## 3 References

The following documents contain provisions, which through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a PoC document, a non-specific reference implicitly refers to the latest version of that document in the same Release as the present document.

[1] “Push-to-talk over Cellular (PoC) User Requirements; PoC Release 2.0”

[2] “Push-to-talk over Cellular (PoC) Architecture; PoC Release 2.0”

[3] “Push-to-talk over Cellular (PoC) Signaling Flows – UE to Network Interface (UNI); PoC Release 2.0”

[4] “Push-to-talk over Cellular (PoC) Signaling Flows – Network to Network Interface (NNI); PoC Release 2.0”

[5] “Push-to-talk over Cellular (PoC) User Plane; Transport Protocols; PoC Release 2.0”

[6] IETF RFC 2396: “Uniform Resource Identifiers (URI): Generic Syntax”.

- [7] IETF RFC 2616: “Hypertext Transfer Protocol HTTP/1.1”.
- [8] IETF RFC 3261: “SIP: session initiation protocol”.
- [9] IETF RFC 2617: “An Extension to HTTP: Digest Access Authentication”.
- [10] IETF RFC 2806: “URLs for Telephone Calls”
- [11] IETF RFC 2279: “UTF-8, a transformation format of ISO 10646”

---

## 4 Definitions and abbreviations

### 4.1 Definitions

For the purposes of the present document, the terms and definitions that are defined in reference [2] apply.

### 4.2 Abbreviations

For the purposes of this document, the abbreviations that are explained in reference [2] apply.

### 4.3 Requirement vocabulary

Shall	Indicates a mandatory requirement.
Should	Indicates a recommendation.
May	Indicates an optional requirement.

---

## 5 List management

In this specification three types of persistent lists are managed for PoC, all of which are stored on the Group and List Management Server (GLMS):

- **Contact lists** are used for storing contact entries in the GLMS server.
- **Group lists** are used to define PoC specific groups.
- **Access lists** are used to define access rules, that is who is allowed or not allowed to reach a specific user via PoC service or presence service.

Access lists are used by the PoC Server. Contact lists and Group lists are used by both the PoC Server and the UE.

Each of the lists stored in the GLMS is assigned to an individual user who is the owner of the list.

NOTE: Besides persistent lists there are also transient lists created for the duration of the talk session that are kept internally on the PoC server and are not stored on the GLMS server. Therefore they are not managed over the Im interface.

### 5.1 Contact list management

A contact list is used by the end user as a means of organizing the identities of other end users, groups, and contact lists. A contact list may be used to address users when initiating PoC communication. Contact lists of type “user” may be used as presence lists.

Contact list management includes operations that allow the user equipment to reliably create, store, modify, retrieve, and delete the contact lists located in the GLMS server as well as to add and remove end user and group identities to/from the list and add and remove contact lists themselves.

The end user may have zero or more contact lists defined.

For each PoC user the GLMS shall maintain a document enumerating all contact lists of the user; the document shall have an attribute "timestamp" that is used for the caching mechanism as described in chapter 7.3.5.

NOTE: The document enumerating contact lists may be constructed when required (e.g. from a database) and does not have to be stored explicitly anywhere. In contrary, the timestamp has to be stored explicitly and shall be modified whenever the end user modifies the list of contact lists, e.g. by adding or deleting a contact list. The timestamp shall not be modified when modifying a contact list or adding/deleting an entry to/from a contact list.

Contact lists are addressable entities on the Im and Is interface. On Im interface the identity of the contact list is included inside the request URI of the HTTP GET request as defined in section 7.3.2, i.e. it is not used as the request URI itself.

### 5.1.1 Contact list attributes

Besides the implicit attribute of the owner the contact list has the following attributes:

- Contact list identity
- Display name
- Type (user or group)
- Default (Yes/No)
- Timestamp
- List of user or group identities; each identity is a tuple containing:
  - URI
  - Display name (optional)

The URI of the entry in the contact list shall be specified as "*scheme:scheme-specific-part*" (see [6]). Each display name shall be represented as an UTF8-encoded UNICODE string (see reference [11]). The GLMS shall allow the URI to be both SIP URI (see reference [8]) and TEL URI (see reference [10]).

The tuples within the list of user or group identities shall be uniquely identified by their URI part. No two tuples in the list shall have the same URI.

The type attribute specifies the type of all entries within the list. The user stores his user contacts to lists of type "user" and his group contacts to lists of type "group". All entries within a single list shall have the same type. The presence service shall use only contact lists of type "user".

The timestamp is used in order to make caching of lists possible on the UE, see chapter 7.3.5 for details on how it is used.

The default attribute is used to enable the UE to identify the user's default contact list. If the value of this attribute is "Y" then the contact list is the default one, otherwise the value of this attribute is "N" and the contact list is not the default one. The usage of the default attribute is UE specific. The GLMS shall ensure that there is only one default contact list for the user. The way in which the UE presents the default attribute to the end user is implementation defined.

NOTE: The UE is not required to use the default attribute at all. The management protocol is designed in such way that the attribute can be omitted in the management operations. The UE shall be aware that the GLMS will include the attribute to the XML document when returning data to the UE.

### 5.1.2 Contact list identities

A contact list is uniquely identified by its SIP URI. This SIP URI is generated by the GLMS when the user creates the contact list. The mechanism how the SIP URI is generated is out of scope of this document.

### 5.1.3 Contact list operations

The GLMS shall support two types of operations on the contact lists of a user:

- Manipulation of contact lists
  - Get a list of contact lists
  - Create a new contact list
  - Delete a contact list
  - Modify contact list attributes
- Manipulation of identities in a contact list
  - Get a list of identities
  - Add an identity to a list
  - Delete an identity from a list
  - Modify identity attributes

### 5.1.4 Contact list policies

By creating a contact list, the creator becomes its owner. Only the owner of the contact list is allowed to manipulate it.

When the UE stores or adds a new identity in the contact list, the GLMS shall not validate that the identity represents an existing entity. It shall however validate that the entity's SIP URI or TEL URL is syntactically valid. After checking that the SIP URI is syntactically valid, the GLMS shall reject the operation if it has an explicit knowledge that the type of the entity referenced by the SIP URI does not match the type of the contact list and shall accept it otherwise.

NOTE: The GLMS may have an explicit knowledge about the type of the entity represented by the SIP URI for example for groups that belong to the same domain or that are stored in the GLMS but may not have knowledge about the type of the entity represented by the SIP URI belonging to other domain.

## 5.2 Access list management

Access list management includes operations that allow the user equipment to reliably manipulate the access lists located in the GLMS and the related attributes.

The access list may be used in PoC and various other services. Each entry in an access list has an attribute describing for which services the entry is valid. The attribute is called "services". The following description describes the generic behavior and details the PoC usage of the access lists.

An access list may be used by the end user as a means of controlling the incoming talk session requests from other users or groups. An access list may contain identities of other users and groups.

The access lists may be applied on the sending and delivery of the instant personal alerts. See [3] for details on instant personal alerts.

There are two access lists per user: a user reject list and a user accept list. Both of these lists may contain a wildcard – "\*" (meaning all other identities). Each URI can be listed in both user accept list and user reject list simultaneously. GLMS should check that the set of services for which an entry is valid in user reject list and the set of services for which the entry (with the same URI) is valid in user accept list are disjoint.

Using the asterisk the user can specify a default policy for his access lists. The default policy can differ service by service; this can be set by using the services attribute. If the asterisk is stated on the user reject list, talk session requests from unlisted subscribers shall be rejected. If the asterisk is stated on the user accept list, session requests sent by unlisted subscribers will not be rejected. If the asterisk is neither on the user reject list nor on the user accept list, the requests are handled as defined in chapter 5.2.5

Wildcards other than above-mentioned asterisk are not allowed.

The access lists are activated or deactivated by the setting of an attribute named “in use”. The value of the attribute shall be either “Y” (for yes, used) or “N” (for no, not used). The “in use” attribute is a common attribute of both access lists and is not attribute of each of the lists separately. The GLMS shall store together with the “in use” attribute also its timestamp, which is used for the caching mechanism described in chapter 7.3.5.

The access lists shall be checked if the access lists are in use (as shown in Figure 1), and shall not be checked otherwise.

NOTE: Access list management includes the “Do-not-Disturb” flag, as explained in the Chapter 6. The Do-not-Disturb flag takes precedence over the access lists. When “Do-not-Disturb” is on, the access lists are not checked.

## 5.2.1 Access list attributes

Besides the implicit attribute of the owner and the type of the list (accept/reject) each of the access lists has the following attributes:

- Timestamp
- List of user or group identities is a list of tuples, each tuple containing:
  - URI or wildcard
  - Display name (optional)
  - Services (optional)

The URI of the entry in the access list shall be specified as "*scheme:scheme-specific-part*" (see [6]). The wildcard shall be represented as "\*" (one single asterisk). Since "\*" is not a valid URI, it's possible to distinguish wildcard from URI. The GLMS shall allow the URI to be both SIP URI (see [8]) and TEL URI (see [10]). The wildcard "\*" shall have no display name, the display name for "\*" shall not be stored in the GLMS.

The tuples within a single access list shall be uniquely identified by their URI part. No two tuples in the list shall have the same URI.

Display name shall be represented as an UTF8-encoded UNICODE string (see [11]).

Services shall be specified as a string of service names; services are delimited by on space. The service names are not case sensitive. The service names shall use only alphanumeric characters.

The service name for PoC shall be “PoC”. The service name for presence service shall be “presence”. Other services shall define their service names for themselves.

The timestamp is used in order to make caching of lists possible on the UE, see chapter 7.3.5 for details on how it is used.

## 5.2.2 Access list identities

The access lists have no identity; they are addressed on the Im interface implicitly using the protocol described in this specification.

## 5.2.3 Access list operations

The GLMS shall support the following operations on the access lists:

- Get the identities in the user accept list/user reject list
- Add identity to the user accept list/user reject list
- Remove identity from the user accept list/user reject list
- Modify attributes of an identity in the user accept list/user reject list
- Set the value of the “in use” attribute

- Get the value of the “in use” attribute

## 5.2.4 Access list policies

Only the owner of the lists is allowed to manage the lists and the “In Use” attribute.

When the UE stores or adds a new identity in the access list, the GLMS shall not validate that the identity represents an existing entity. It shall however validate that the entity’s SIP URI or TEL URL is syntactically valid.

## 5.2.5 Processing of access lists

When a service consults the access lists it only takes into account the entries of the access lists that are valid for the service, in other words only those for which the service is either explicitly listed in the “services” attribute of the entry or those for which the entry is empty. The processing of the access lists is done as shown in Figure 1 and Figure 2.

For PoC the access lists are consulted every time the PoC server is requested to add a participant to a talk session or when a PoC user subscribes for participant information in a talk session in which he does not participate. The processing of access lists includes checking the “in use” attribute, locating the wildcard in the access lists and consulting other access list entries. The access lists shall be matched against the identity of the initiator of the talk session request or subscription request; that is the SIP URI of the user and/or group that initiated the request.

The processing of access lists when handling talk session request is shown in Figure 1, processing of access list when handling subscription for participant information in a talk session in which the initiator of the request does not participate is shown in Figure 2.

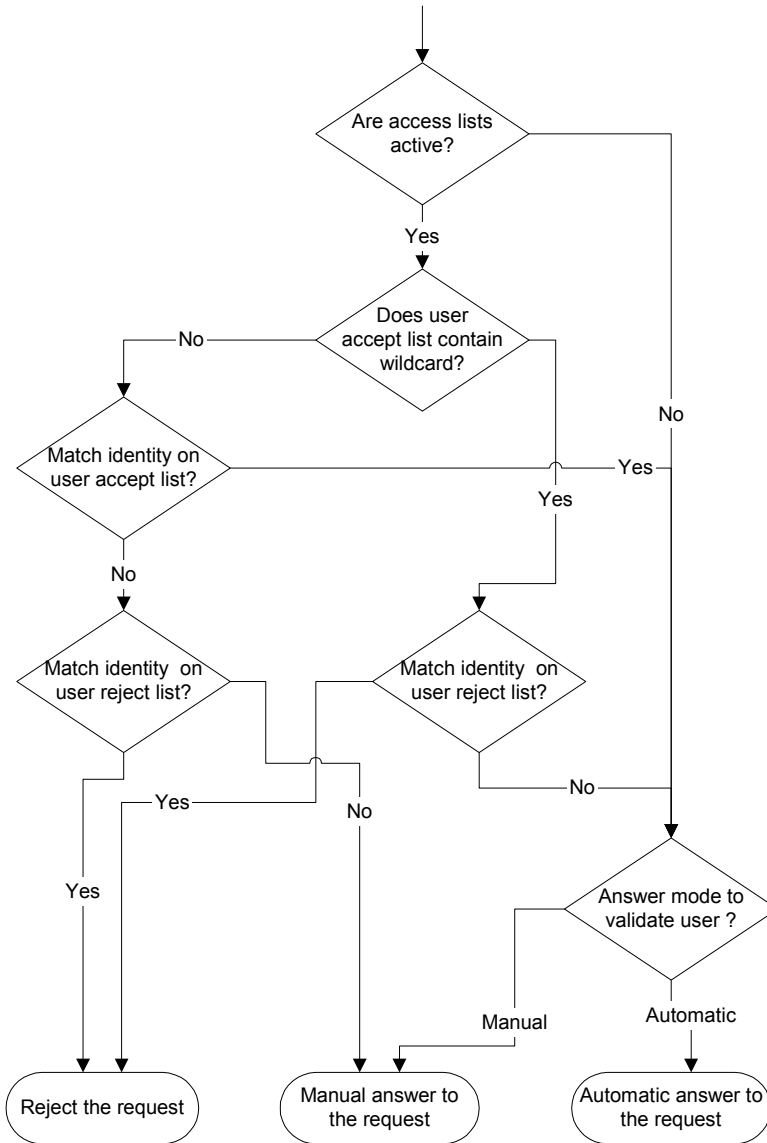


Figure 1: Processing of access lists upon incoming talk session request.

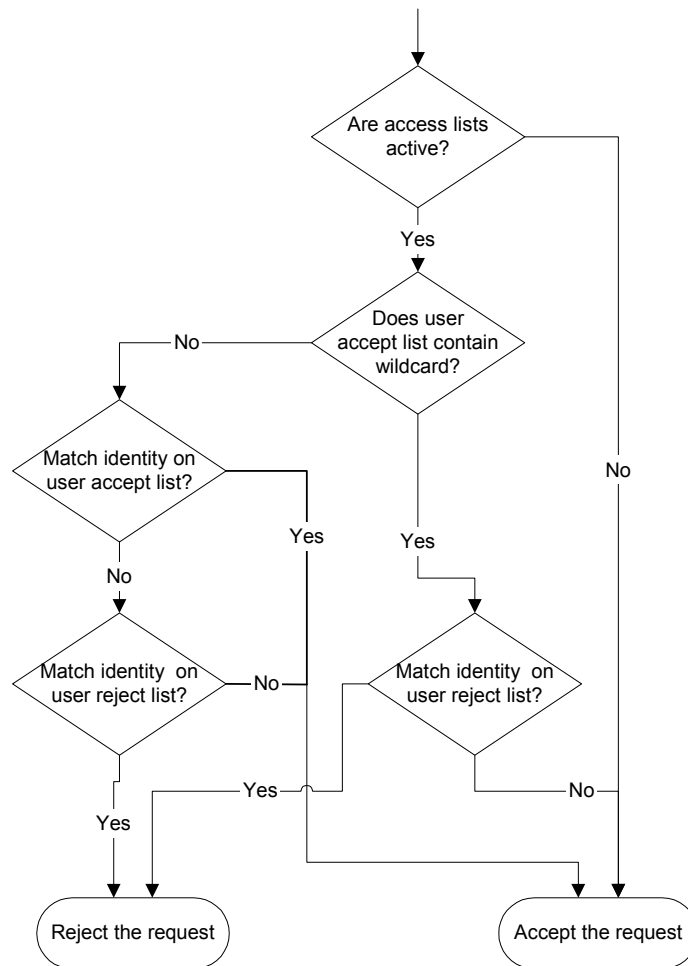


Figure 2: Processing of access lists upon incoming participant information subscription request.

## 5.3 Group list management

Group management includes operations that allow the user equipment to reliably store and retrieve the groups located in the GLMS server as well as to create and delete the groups and change their attributes. This includes manipulation of lists that are part of the group definition.

The end user uses a group as a means to establish talk sessions. The group attributes control the session type and who may participate in the talk session. The group identity is used on the Is interface to address the group and initiate group talk session (for more details about Im – Is correlation see the chapter 8).

The end user may have zero or more groups defined.

For each PoC user the GLMS shall maintain a document enumerating all groups of the user; the document shall have an attribute "timestamp" that is used for the caching mechanism as described in chapter 7.3.5.

**NOTE:** The document enumerating groups may be constructed when required (e.g. from a database) and does not have to be stored explicitly anywhere. In contrary, the timestamp has to be stored explicitly and shall be modified whenever the end user modifies the list of groups, e.g. by adding or deleting a group. The timestamp shall not be modified when modifying a group or adding/deleting an entry to/from a group member list or group reject list.

### 5.3.1 Group attributes

Besides the implicit attribute of the owner, the group has the following attributes:

- Display name
- Group Identity
- Timestamp
- Talk session type (chat/instant)
- Membership (open/restricted)
- Group reject list is a list of tuples, each tuple containing:
  - URI
  - Display name (optional)
- Group member list is a list of tuples, each tuple containing:
  - URI
  - Display name (optional)
- Maximum number of participants
- Anonymous access
- Owner initiated only
- Owner add only

The URI of entries in group reject list and group member list shall be specified as "*scheme:scheme-specific-part*" (see [6]). The GLMS shall allow the URI to be both SIP URI (see [8]) and TEL URI (see [10]). Each display name shall be represented as an UTF8-encoded UNICODE string (see [11]).

The tuples within the group reject list and group member list shall be uniquely identified by their URI part. No two tuples in the lists shall have the same URI. GLMS should check that same URI is not listed to both accept and reject lists. Talk session type defines whether the talk session is of type “chat” or “instant” talk. For instant talk all the group members are invited to the talk session on the end user’s request (valid only for a restricted group), while chat implies that each group member joins the talk session individually.

Membership shall be set either to “open” or to “restricted”.

NOTE: For open, any end user may join the group, while restricted implies that only particular end users on the member list may participate in the group talk session. The group reject list shall be used to restrict access to open groups.

Group reject list and group member list contain user identities (SIP URI or TEL URL). Group identities shall not be included in any of these lists.

The maximum number of participants attribute specifies the maximum number of participants that may be concurrently engaged in a group talk session. The network operator may impose a limit on the maximum number of participants based on local policy.

The anonymous access attribute allows setting that the PoC server shall hide public user identities of the session participants from other participants. The value of the anonymous attribute shall be either “Y” or “N”. The PoC server shall hide identities of all participants when the value is set to “Y” and shall not hide identity of any user otherwise.

The anonymous access attribute shall be only valid for chat groups; it shall have no effect on instant groups.

The owner initiated only attribute allows the owner of the group to set that only he is allowed to start the group talk or if anybody who is granted access to the group can initiate the group talk. The value of the owner initiated only attribute shall be either “Y” or “N”. The PoC server shall allow all the users that are granted access to the group according to group reject list and group member list to initiate the group talk if and only if the value of this parameter is “N”. The owner of the group can initiate group talk regardless of the value of the owner initiated only parameter.

The owner add only attribute allows the owner of the group to set that only he is allowed to invite participant to the group talk or if anybody is allowed to invite participants to the group talk. The value of the owner add only attribute shall be either “Y” or “N”. The PoC server shall allow the users that are participants of the group talk to add users to the group talk if and only if the value of this parameter is “N”. The owner of the group can add users to group talk regardless of the value of the owner add only parameter.

The timestamp is used in order to make caching of lists possible on the UE, see chapter 7.3.5 for details on how it is used.

### 5.3.2 Group identities

A group is uniquely identified by a SIP URI. This SIP URI is generated by the GLMS when the user creates the group. The mechanism how the URI is generated is out of scope of this document.

### 5.3.3 Group operations

The GLMS shall support two types of operations on the groups of a user:

- Manipulation of groups
  - Get a list of groups
  - Create a group
  - Delete a group
- Manipulation of group attributes
  - Modification of attribute (talk session type, membership, anonymous access, owner initiated only, owner add only)
  - Manipulation of group reject list and group member list
    - Get entries in the list
    - Add an entry to the list
    - Delete an entry from the list
    - Modify entry attributes

### 5.3.4 Group policies

Group policy is applied on the Im interface.

By creating a group, the creator becomes its owner. Only the owner of the group is allowed to manipulate it.

When the UE stores or adds a new identity in the group member list or group reject list of a group, the GLMS shall not validate that the identity represents an existing entity. It shall however validate that the entity’s SIP URI or TEL URL is syntactically correct. After checking that the SIP URI is syntactically correct, the GLMS shall reject the operation if it has an explicit knowledge that the type of the entity referenced by the SIP URI is not end user and shall accept it otherwise.

**NOTE:** The GLMS may have an explicit knowledge about the type of the entity represented by the SIP URI for example for groups that belong to the same domain or that are stored in the GLMS but may not have knowledge about the type of the entity represented by the SIP URI belonging to other domain.

## 6 Do-not-Disturb

Do-not-Disturb is a PoC feature that allows the end user to block all incoming talk session requests. The Do-not-Disturb flag shall have no effect on the sending and delivery of the talk alerts. See [3] for details on talk alerts.

The DnD feature uses list management operations to achieve the desired behavior of blocking all incoming talk session requests addressed to the end user in the PoC Server. This behavior is achieved by setting a DnD flag in the GLMS.

### 6.1 Do-not-Disturb attribute

The name of the attribute is “Do-not-Disturb”. Every user shall own one Do-not-Disturb flag stored on the GLMS. The Do-not-Disturb flag shall always have value “Y” or “N”.

### 6.2 Do-not-Disturb identity

The Do-not-Disturb flag has no identity; it is addressed on the Im interface implicitly using the protocol described in this specification.

### 6.3 Do-not-Disturb operations

The GLMS shall support the following operations on the Do-not-Disturb flag:

- Set the value of the “Do-not-Disturb” flag
- Get the value of the “Do-not-Disturb” flag

After power on the UE should fetch the Do-not-Disturb setting from the server to avoid any synchronization inconsistencies.

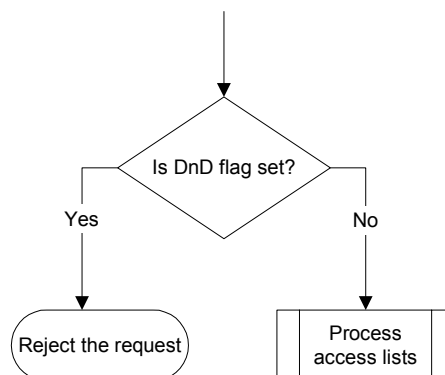
### 6.4 Do-not-Disturb policies

Policy for Do-not-Disturb flag shall always allow the owner to query and set the value of this flag. Users shall be rejected when attempting to set the value of the Do-not-Disturb flag that they do not own.

### 6.5 Processing of Do-not-Disturb flag

If Do-not-Disturb flag is set to “Y”, then the PoC server shall block all incoming talk requests to the owner. If this flag is set to “N”, then the PoC server shall block the incoming talk requests according to the access lists of the same owner.

Meanwhile after the user through his UE requests activation of the don-not-disturb flag and before the server applies the setting and confirms it to the UE the UE may automatically reject all incoming talk requests.



**Figure 3: Processing of DnD flag upon incoming talk session request. See section 5.2.5 for details on how the access lists are processed.**

## 7 Description of the list management protocol

### 7.1 General protocol requirements

The list management protocol requires that downloading data from the GLMS server to the UE is possible in order to modify the data.

PoC list management on the Im interface shall use the HTTP Version 1.1 as defined in RFC2616 [7].

The UE shall request the action in a HTTP GET request. The action and its parameters shall be described in the HTTP URL. The status of the action (passed/failed) shall be returned in the HTTP response status code. The action may return some output parameters; they shall be returned in the form of XML in the body of HTTP response (each action has a specific content type and associated XML structure assigned).

The input parameters may be UNICODE strings. Since HTTP is based on 8 bit strings, the input parameters in the HTTP URL shall be encoded using UTF-8 (and escaped if necessary) (see [11]).

The address of the HTTP server, which provides the GLMS functionality, shall be a UE configuration parameter (the GLMS Address parameter).

The list management protocol on Im interface is bandwidth efficient through versioning of the lists on the server and conditional downloading of the lists only if the version has changed.

### 7.2 Authentication

The list management protocol shall follow the HTTP Digest authentication as specified in [9] with the following clarifications pertinent to this specification:

- MD5 checksum shall be used;
- The UE shall include its credentials preemptively in Authorization header of each request to avoid unnecessary round trips for authentication challenge;
- The use of authentication with integrity protection (“auth-int”) is recommended;
- The UE shall support both HTTP server and HTTP proxy authentication.

Each HTTP action using HTTP Digest authentication shall have the “owner” parameter in the Request-URI.

Parameter	Mandatory / Optional	Purpose
Owner	M	The Public User Identity of the owner, which is a SIP URI of the owner of the manipulated object

For example:

```
GET http://glms.operator.com/script?action=create_buddy_list&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_displayname=Colleagues HTTP/1.1
```

Every time the UE sends a request it shall preemptively send its credentials based on the information previously received from the GLMS server (if any). The GLMS may challenge any request from the UE by responding with a "401 Unauthorized" with "WWW-Authenticate" header comprising following parameters:

Parameter	Mandatory / Optional	Purpose
Realm	M	The operator's domain
Domain	O	A list of URIs defining the protection space
Nonce	M	A unique string specified by the server side
Opaque	O	A string of data, specified by the server, which should be returned by the UE unchanged
Stale	O	Used to indicate stale credentials
Algorithm	O	If present, the value of "MD5" shall be used
Qop	O	"auth-int" is recommended

For example:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm="glms.umts.uk",
    nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
    opaque="5ccc069c403ebaf9f0171e9517f40e41",
    qop=auth-int
```

The UE shall use the Private User Identity and the user password configured in the UE to generate the user credentials in a request. The request shall include the "Authorization" header indicating Digest authentication scheme and comprising the following parameters:

Parameter	Mandatory / Optional / Conditional	Purpose
username	M	The Private User Identity (only the user part)
Realm	M	The operator's domain
Nonce	M	A string of data as specified by the server
Uri	M	The URI from the Request-URI
response	M	MD5 checksum generated by the UE
Qop	C	Shall be sent if the received challenge had the directive; "auth-int" is recommended
Cnonce	C	Shall be sent if qop is included
Nc	C	Shall be sent if qop is included
opaque	C	Shall be sent if received from the server; a string of data as specified by the server

For example:

```
GET http://glms.operator.com/script?action=create_buddy_list&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_displayname=Colleagues HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
    nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
    uri="http://glms.operator.com/script?action=create_buddy_list&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_displayname=Colleagues",
    response="e966c932a9242554e42c8ee200cec7f6",
    cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
    opaque="5ccc069c403ebaf9f0171e9517f40e41",
    qop=auth-int,
    nc=00000001
```

## 7.3 Protocol definition

The following objects and attributes can be managed via the Im interface

- DnD flag
- Answer mode flag
- Contact lists (zero, one or more lists)

- Access lists (one user accept list and one user reject list and “in\_use” attribute)
- Groups (zero, one or more groups)

Note: Examples are not normative.

### 7.3.1 Attribute manipulation

Im operations on the attribute manipulation include:

- Set the value of -flag
- Get the value of flag

#### 7.3.1.1 Set attributes

This method shall change per-user attributes.

Parameter	Value	Purpose
Action	Set_attributes	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Dnd	"Y"   "N"	(optional) A value defining whether DnD mode is "active".
Answer_mode	"manual"   "automatic"	(optional) A value defining the instant personal talk answer mode to validate users.

As the result of invocation of this method, the attributes whose values are present in the parameters of the method invocation are set to their new values. In this specification, two attributes are defined – the Do-not-Disturb and Answer-Mode flags.

The dnd and/or answer\_mode parameter shall be present in the request in order to change its value. If the parameter is not present, the value of the corresponding Flag is not changed.

Note: This behavior allows for extensibility of the protocol and allows reuse of this method for changing various other attributes that might be added in future versions of the specification.

For example:

Ron Underwood wants to reset the DnD flag (HTTP Digest authentication)

```
GET http://glms.operator.com/script?action=set_attributes&owner=sip%3Aronald.underwood%40umts.uk&dnd=N HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=set_attributes&owner=sip%3Aronald.underwood%40umts.uk&dnd=N",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

Ron Underwood wants to set the answer\_mode flag (HTTP Digest authentication)

```
GET http://glms.operator.com/script?action=set_attributes&owner=sip%3Aronald.underwood%40umts.uk&answer_mode=automatic HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=set_attributes&owner=sip%3Aronald.underwood%40umts.uk&answer_mode=automatic",
```

```

response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001

```

HTTP/1.1 200 OK

Ron Underwood wants to set both dnd and answer\_mode flags (HTTP Digest authentication)

```

GET http://glms.operator.com/script?action=set_attributes&owner=sip%3Aronald.und
erwood%40umts.uk&dnd=Y&answer_mode>manual HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=set_attributes&owner=sip%3Arona
ld.underwood%40umts.uk&answer_mode>manual",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001

```

HTTP/1.1 200 OK

### 7.3.1.2 Get attributes

This method shall return a list of per-user attributes and their values, which are valid for the user (owner).

Parameter	Value	Purpose
Action	get_attributes	The type of action to be done.
Owner		A SIP URI of the user performing the action.

As the result of invocation of this method, a document is returned in the HTTP “200 OK” final response with content type “application/attributes+xml” (see sub-clause 7.5.7) encoded as UTF-8 (see reference [11]), which describes the values of all per-user attributes.

For example:

Ron Underwood wants to get the attributes (HTTP DIGEST authentication)

```

GET http://glms.operator.com/script?action=get_attributes&owner=sip%3Aronald.und
erwood%40umts.uk HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="
http://glms.operator.com/script?action=get_attributes&owner=sip%3Aronald.un
derwood%40umts.uk",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001

```

HTTP/1.1 200 OK

Content-Type: application/attributes+xml;charset=UTF-8

```

<?xml version="1.0" encoding="UTF-8"?>
<attributes>
  <attribute name="dnd">Y</attribute>
  <attribute name="answer_mode">>manual</attribute>
</attributes>

```

## 7.3.2 Contact lists

Im operations for Contact Lists include

- Create/delete/modify Contact List
- Add/delete/modify user/group identity
- Store/Retrieve Contact List (storing is done implicitly by other operations)
- Get list of contact lists
- Set the “default” attribute
- Get the “default” attribute

Wildcards are not allowed in the contact lists.

### 7.3.2.1 Create Contact list

This method shall create a new empty contact list. The GLMS shall generate a unique SIP URI for the contact list and shall return it to the UE in the success response to this request. The actual SIP URI generation mechanism is implementation defined.

Parameter	Value	Purpose
Action	create_buddy_list	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Buddy_list_displayname		Display name of the contact list being added.
default	"Y"   "N"	(optional) Specifies whether this is the default contact list.
Type	"user"   "group"	(optional) Type of the list. If not set the default “user” is assumed.

As the result of invocation of this method, the GLMS shall create a new contact list for the owner, shall generate a unique SIP URI for it, shall set the list attributes to the requested values and shall return in the HTTP “200 OK” final response a document of content type application/buddylist+xml (see sub-clause 7.5.1) encoded as UTF-8 (see reference [11]) that describes the created contact list. The description shall contain the generated SIP URI of the contact list.

The value of buddy\_list\_displayname must not be an empty string.

The “default” parameter is optional. When not present, the contact list shall be set to non-default. When inserting the first contact list, the GLMS shall make it always the default regardless of the value of this parameter. If present and set to “Y”, the contact list shall be set to default and the contact list that was default before this operation shall be marked non-default.

The type parameter is optional. When not present, the semantics shall be the same as if it was present and set to type “user”.

For example, the URI can be generated based on the SIP URI of the owner and the display name of the contact list as shown in the following example. Various other ways of generating the URI are possible. In the example below the presence of “contact-list” tag in the SIP URI is not required when using other URI generation mechanisms.

For example:

Ron Underwood wants to create a contact list named "Colleagues" (HTTP Digest authentication)

```
GET http://glms.operator.com/script?action=create_buddy_list&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_displayname=Colleagues HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=create_buddy_list&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_displayname=Colleagues",
response="e966c932a9242554e42c8ee200cec7f6",
```

```

cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001

```

HTTP/1.1 200 OK

Content-Type: application/buddylist+xml;charset=UTF-8

```

<?xml version="1.0" encoding="UTF-8"?>
<buddylist uri="sip:ronald.underwood@umts.uk;contact-list=Colleagues"
  displayname="Colleagues" default="N" type="user"/>

```

### 7.3.2.2 Delete Contact List

This method shall delete an existing contact list.

Parameter	Value	Purpose
Action	delete_buddy_list	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Buddy_list_uri		URI of the contact list to be deleted.

As the result of invocation of this method, the contact list identified by buddy\_list\_uri shall be deleted including all entries on the list.

If the default contact list is deleted and there are still contact lists stored on the GMLS, the server may choose any of these contact list and may make it the default contact list. The mechanism of choosing the new default contact list is implementation defined.

For example:

Ron Underwood wants to delete his contact list named "Colleagues" (HTTP Digest authentication)

```

GET http://glms.operator.com/script?action=delete_buddy_list&owner=sip%3Aronald.
  underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40umts.uk%3Bconta
  ct-list%3dColleagues HTTP/1.1

```

```

Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=delete_buddy_list&owner=sip%3Ar
  onald.underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40umts.uk%3
  Bcontact-list%3dColleagues",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001

```

HTTP/1.1 200 OK

### 7.3.2.3 Add user/group identity to contact list

This method shall insert a new user or a group to a contact list.

Parameter	Value	Purpose
Action	add_buddy	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Buddy_list_uri		URI of the contact list to be modified.
Buddy_uri		URI of user or group to be added.
Buddy_displayname		(optional) Display name of user or contact list or group being added.

As the result of invocation of this method, the user or group identified in the `buddy_uri` parameter shall be inserted to the contact list identified in the `buddy_list_uri` parameter.

If the `buddy_displayname` parameter is not present or its value is an empty string then the new entry in the contact list shall have no display name.

For example:

Ron Underwood wants to add `george.underwood@umts.uk` into his contact list named "Colleagues" (HTTP Digest authentication)

```
GET http://glms.operator.com/script?action=add_buddy&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40umts.uk%3Bcontact-list%3dColleagues&buddy_uri=sip%3Ageorge.underwood%40umts.uk HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=add_buddy&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40umts.uk%3Bcontact-list%3dColleagues&buddy_uri=sip%3Ageorge.underwood%40umts.uk",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

### 7.3.2.4 Delete user/group identity from contact list

This method shall delete a user or a group from the contact list.

Parameter	Value	Purpose
Action	delete_buddy	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Buddy_list_uri		URI of the contact list to be modified.
Buddy_uri		URI of user or or group to be deleted.

As the result of invocation of this method, the user or group identified by the `buddy_uri` parameter shall be deleted from the contact list identified by the `buddy_list_uri` parameter. The identity being deleted must already exist in the contact list.

For example:

Ron Underwood wants to remove `dean.burrows@umts.uk` from his contact list named "Colleagues" (HTTP DIGEST authentication)

```
GET http://glms.operator.com/script?action=delete_buddy&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40umts.uk%3Bcontact-list%3dColleagues&buddy_uri=sip%3Adean.burrows%40umts.uk HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=delete_buddy&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40umts.uk%3Bcontact-list%3dColleagues&buddy_uri=sip%3Adean.burrows%40umts.uk",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

### 7.3.2.5 Retrieve Contact List

This method shall serve for retrieving the contents of the individual contact lists.

Parameter	Value	Purpose
Action	get_list_of_buddies	The type of action to be done.
Owner		A SIP URI of the user performing the action.
buddy_list_uri		URI of the contact list whose content is to be retrieved.

As the result of invocation of this method, the contents of the contact list identified by the `buddy_list_uri` parameter shall be returned in the body of the HTTP “200 OK” final response. The content-type of the response shall be “application/buddies+xml” (see sub-clause 7.5.3) and shall use UTF-8 encoding (see reference [11]).

The optional attribute “services” shall not be present in the response.

For example:

Ron Underwood wants to retrieve content of his contact list named "Colleagues" (HTTP Digest authentication).

```
GET http://glms.operator.com/script?action=get_list_of_buddies&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40umts.uk%3Bcontact-list%3dColleagues HTTP/1.1
```

```
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=get_list_of_buddies&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40umts.uk%3Bcontact-list%3dColleagues",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

```
HTTP/1.1 200 OK
```

```
Content-Type: application/buddies+xml;charset=UTF-8
```

```
<?xml version="1.0" encoding="UTF-8"?>
<buddies type="user">
  <buddy displayname="Hermione" uri="sip:hermione.blossom@umts.uk"/>
  <buddy uri="sip:george.underwood@umts.uk"/>
</buddies>
```

Ron Underwood wants to retrieve content of his contact list named "My Company" (HTTP DIGEST authentication).

```
GET http://glms.operator.com/script?action=get_list_of_buddies&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40umts.uk%3Bcontact-list%3dMy_Company HTTP/1.1
```

```
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=get_list_of_buddies&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40umts.uk%3Bcontact-list%3dMy_Company",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

```
HTTP/1.1 200 OK
```

```
Content-Type: application/buddies+xml;charset=UTF-8
```

```
<?xml version="1.0" encoding="UTF-8"?>
<buddies type="group">
```

```
<buddy displayname="Sales" uri="sip:ron.underwood@umts.uk;group=Sales"/>
<buddy displayname="Marketing"
  uri="sip:ron.underwood@umts.uk;group=Marketing"/>
</buddies>
```

### 7.3.2.6 Modify user/group attributes in contact list

This method shall modify the attributes of an entry in the contact list.

Parameter	Value	Purpose
Action	modify_buddy	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Buddy_list_uri		URI of the contact list to be modified.
Buddy_uri		URI of user or group to be modified.
Buddy_displayname		(optional) New display name of user, contact list or group.

As the result of invocation of this method, the GLMS shall set the values of the attributes of the entry identified by the buddy\_uri parameter in the contact list identified by the buddy\_list\_uri parameter to their new values requested. If an optional parameter is missing, the GLMS shall not modify its value.

If the value of buddy\_displayname parameter is an empty string then the modified entry in the contact list shall have no display name after the method is invoked.

Note: This behavior allows for extensibility of the protocol and allows reuse of this method for changing various other attributes that might be added in future versions of the specification.

For example:

Ron Underwood wants to change display name of dean.burrows@umts.uk in the contact list named "Colleagues" (HTTP DIGEST authentication)

```
GET http://glms.operator.com/script?action=modify_buddy&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40umts.uk%3Bcontact-list%3dColleagues&buddy_uri=sip%3Adean.burrows%40umts.uk&buddy_displayname=Supporter HTTP/1.1
```

```
Authorization: Digest username="u29502566", realm="glms.umts.uk",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
  uri="http://glms.operator.com/script?action=modify_buddy&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40umts.uk%3Bcontact-list%3dColleagues&buddy_uri=sip%3Adean.burrows%40umts.uk&buddy_displayname=Supporter",
  response="e966c932a9242554e42c8ee200cec7f6",
  cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
  opaque="5ccc069c403ebaf9f0171e9517f40e41",
  qop=auth-int,
  nc=00000001
```

HTTP/1.1 200 OK

### 7.3.2.7 Get list of contact lists

This method shall retrieve a list of all contact lists belonging to the subscriber (owner).

Parameter	Value	Purpose
Action	get_list_of_buddy_list	The type of action to be done.
Owner		A SIP URI of the user performing the action.

As the result of invocation of this method, the GLMS shall return in HTTP "200 OK" final response a document of content type "application/buddylists+xml" (see sub-clause 7.5.2) encoded as UTF-8 (see reference [11]) describing all the contact lists that belong to the invoking subscriber.

For example:

Ron Underwood wants to retrieve a list of all his contact lists (HTTP DIGEST authentication)

```
GET http://glms.operator.com/script?action=get_list_of_buddy_list&owner=sip%3Aronald.underwood%40umts.uk HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
  uri="http://glms.operator.com/script?action=get_list_of_buddy_list&owner=sip%3Aronald.underwood%40umts.uk",
  response="e966c932a9242554e42c8ee200cec7f6",
  cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
  opaque="5ccc069c403ebaf9f0171e9517f40e41",
  qop=auth-int,
  nc=00000001
```

```
HTTP/1.1 200 OK
Content-Type: application/buddylists+xml;charset=UTF-8
```

```
<?xml version="1.0" encoding="UTF-8"?>
<buddylists>
  <buddylist uri="sip:ronald.underwood@umts.uk;contact-list=Colleagues"
    displayname="Colleagues" default="N" type="user"/>
  <buddylist uri="sip:ronald.underwood@umts.uk;contact-list=Family"
    displayname="Family" default="Y" type="user"/>
  <buddylist uri="sip:ronald.underwood@umst.uk;contact-list=Groups"
    displayname="Groups" type="group">
</buddylists>
```

### 7.3.2.8 Modify contact list attributes

This method shall modify contact list attributes.

Parameter	Value	Purpose
Action	set_buddy_list_attributes	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Buddy_list_uri		URI of the contact list to be modified.
Buddy_list_displayname		(optional) New display name of the contact list.
Default	"Y"   "N"	(optional) Specifies whether this is the default contact list.

As the result of invocation of this method, the GLMS shall set the attributes of an existing contact list identified by `buddy_list_uri` to their new values as request in the method parameters. If the optional parameter is not present in the request, then the value of the corresponding attribute shall not be changed. If attribute value update executed successfully the GLMS shall respond to the request with an HTTP "200 OK" final response.

The value of `buddy_list_displayname` parameter must not be an empty string.

Note: This behavior allows for extensibility of the protocol and allows reuse of this method for changing various other attributes that might be added in future versions of the specification.

Note: Changing of the "type" attribute is not possible.

As in chapter 7.3.2.1 in case that the parameters specified by the user were overridden by the server the HTTP "200 OK" response to this request shall contain body of type "application/buddylist+xml" (see subclause 7.5.1) encoded as UTF-8 (see reference [11]) describing the values of the overridden parameters.

For example:

Ron Underwood wants to set the contact list to be default (HTTP Digest authentication)

```
GET http://glms.operator.com/script?action=set_buddy_list_attributes&owner=sip%3Aronald.underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40umts.uk%3Bcontact-list%3DColleagues&default=Y HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
```

```

uri="http://glms.operator.com/script?action=set_buddy_list_attributes&owner
=sip%3Aronald.underwood%40umts.uk&buddy_list_uri=sip%3Aronald.underwood%40u
mts.uk%3Bcontact-list%3DColleagues&default=Y",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001

```

HTTP/1.1 200 OK

Content-Type: application/buddylist+xml; charset=UTF-8

```

<?xml version="1.0" encoding="UTF-8"?>
<buddylist uri="sip:ronald.underwood@umts.uk;contact-list=Colleagues"
displayname="Colleagues" default="Y"/>

```

### 7.3.3 Access Lists

Im operations for Access Lists include

- Add/delete/modify user/group identity
- Store/Retrieve Access List (storing is done implicitly by other operations)
- Modify Access List

Each user has exactly one user accept list and one user reject list. These lists shall exist by default in the GLMS and therefore there are no specific add or delete operations allowed on these lists.

Each entry in an access list may have an attribute specifying for which service the entry is valid. If the value of the attribute is not specified, the entry is valid for all services. This allows for backward compatibility, especially in case when PoC 1.0 terminal uses the PoC 2.0 network.

#### 7.3.3.1 Add user/group identity to user accept list

This method shall insert new entries to the user accept list.

Parameter	Value	Purpose
Action	add_user_accept_list_entry	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Buddy_uri		URI of user or group to be added or "*".
Buddy_displayname		(optional) Display name of user or group being added.
services		(optional) List of service names formatted as described in 5.2.1.

As the result of the invocation of this method the identity identified by buddy\_uri with display name in buddy\_displayname shall be inserted into the user accept list of the owner. The validity of the entry for services shall be set as specified in services parameter.

If buddy\_uri is "\*" then the value in buddy\_displayname shall be ignored. The services attribute shall not be ignored.

If the buddy\_displayname parameter is not present or its value is an empty string then the new entry in the user accept list shall have no display name.

If the services parameter is not present or its value is an empty string then the new entry in the user accept list shall be valid for all services.

For example:

Ron Underwood wants to add dean.burrows@umts.uk into his user accept list (HTTP DIGEST authentication) for PoC and presence.

```

GET http://glms.operator.com/script?action=add_user_accept_list_entry&owner=sip%
3Aronald.underwood%40umts.uk&buddy_uri=sip%3Adean.burrows%40umts.uk&buddy_d
isplayname=Dean+Burrows&services=poc+presence HTTP/1.1

```

```
Authorization: Digest username="u29502566", realm="glms.umts.uk",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
  uri="http://glms.operator.com/script?action=add_user_accept_list_entry&owner=sip%3Aronald.underwood%40umts.uk&buddy_uri=sip%3Adean.burrows%40umts.uk&buddy_displayname=Dean+Burrows&services=poc+presence",
  response="e966c932a9242554e42c8ee200cec7f6",
  cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
  opaque="5ccc069c403ebaf9f0171e9517f40e41",
  qop=auth-int,
  nc=00000001
```

HTTP/1.1 200 OK

### 7.3.3.2 Delete user/group identity from user accept list

This method shall delete entries from user accept list.

Parameter	Value	Purpose
Action	delete_user_accept_list_entry	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Buddy_uri		URI of user or group to be deleted or “*”.

As the result of invocation of this method, the entry identified by buddy\_uri shall be deleted from the user accept list of the owner.

For example:

Ron Underwood wants to delete dean.burrows@umts.uk from his accept list (HTTP Digest authentication)

```
GET http://glms.operator.com/script?action=delete_accept_list_entry&owner=sip%3Aronald.underwood%40umts.uk&buddy_uri=sip%3Adean.burrows%40umts.uk HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
  uri="http://glms.operator.com/script?action=delete_accept_list_entry&owner=sip%3Aronald.underwood%40umts.uk&buddy_uri=sip%3Adean.burrows%40umts.uk",
  response="e966c932a9242554e42c8ee200cec7f6",
  cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
  opaque="5ccc069c403ebaf9f0171e9517f40e41",
  qop=auth-int,
  nc=00000001
```

HTTP/1.1 200 OK

### 7.3.3.3 Modify user/group attributes in user accept list

This method shall modify attributes of entries in the user accept list.

Parameter	Value	Purpose
Action	modify_user_accept_list_entry	The type of action to be done.
Owner		A SIP URI of the user performing the action.
buddy_uri		URI of user or group to be modified (“*” is not allowed here”)
buddy_displayname		(optional) New display name of user or group.
services		(optional) List of service names formatted as described in 5.2.1.

As the result of invocation of this method, the GLMS shall set attributes of the entry identified by buddy\_uri parameter in the user accept list to the values supplied in this request. If an optional parameter is not present in the request, then the corresponding attribute shall not be changed.

If the value of buddy\_displayname parameter is an empty string then the modified entry in the user accept list shall have no display name after the method is invoked.

If the services parameter is an empty string then the modified entry in the user accept list shall be valid for all services.

Note: This behavior allows for extensibility of the protocol and allows reuse of this method for changing various other attributes that might be added in future versions of the specification.

For example:

Ron Underwood wants to change display name of dean.burrows@umts.uk in the accept list (HTTP DIGEST authentication) and wants that dean.burrows@umts is only in accept list for presence service.

```
GET http://glms.operator.com/script?action=modify_user_accept_list_entry&owner=sip%3Aronald.underwood%40umts.uk&buddy_uri=sip%3Adean.burrows%40umts.uk&buddy_displayname=Supporter&services=presence HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=modify_user_accept_list_entry&owner=sip%3Aronald.underwood%40umts.uk&buddy_uri=sip%3Adean.burrows%40umts.uk&buddy_displayname=Supporter&services=presence",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

### 7.3.3.4 Retrieve user accept list

This method shall retrieve the contents of the user accept list.

Parameter	Value	Purpose
Action	get_user_accept_list	The type of action to be done.
Owner		A SIP URI of the user performing the action.

As the result of invocation of this method, the GLMS generates a document of content-type application/buddies+xml (see sub-clause 7.5.3) encoded as UTF-8 (see reference [11]) describing the contents of the user accept list and shall return it in the HTTP "200 OK" final response to the initiator of the request.

For example:

Ron Underwood wants to retrieve content of his useraccept list (HTTP Digest authentication)

```
GET http://glms.operator.com/script?action=get_user_accept_list&owner=sip%3Aronald.underwood%40umts.uk HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=get_user_accept_list&owner=sip%3Aronald.underwood%40umts.uk",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

Content-Type: application/buddies+xml; charset=UTF-8

```
<?xml version="1.0" encoding="UTF-8"?>
<buddies>
  <buddy uri="*" services="poc"/>
  <buddy displayname="Percy" uri="sip:percy.underwood@umts.uk"
    services="poc presence"/>
</buddies>
```

### 7.3.3.5 Add user/group identity to user reject list

This method shall insert new entries to the user reject list.

Parameter	Value	Purpose
Action	add_user_reject_list_entry	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Buddy_uri		URI of user or group to be added or “*”.
Buddy_displayname		(optional) Display name of user or contact list or group being added.
services		(optional) List of service names formatted as described in 5.2.1.

As the result of the invocation of this method the identity identified by buddy\_uri with display name inbuddy\_displayname shall be inserted into the user reject list of the owner. The validity of the entry for services shall be set as specified in services parameter.

If buddy\_uri is “\*” then the value in buddy\_displayname shall be ignored.

If the buddy\_displayname parameter is not present or its value is an empty string then the new entry in the user reject list shall have no display name.

If the services parameter is not present or its value is an empty string then the new entry in the user reject list shall be valid for all services.

For example:

Ron Underwood wants to add tom.nettle@umts.uk into his user reject list (HTTP Digest authentication) for PoC service.

```
GET http://glms.operator.com/script?action=add_user_reject_list_entry&owner=sip%
3Aronald.underwood%40umts.uk&buddy_uri=sip%3Atom.nettle%40umts.uk&services=
poc HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=add_user_reject_list_entry&owne
r=sip%3Aronald.underwood%40umts.uk&buddy_uri=sip%3Atom.nettle%40umts.uk&ser
vices=poc",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

```
HTTP/1.1 200 OK
```

### 7.3.3.6 Delete user/group identity from user reject list

This method shall delete entries from user reject list.

Parameter	Value	Purpose
Action	delete_user_reject_list_entry	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Buddy_uri		URI of user or contact list or group to be deleted or “*”.

As the result of invocation of this method, the entry identified by buddy\_uri shall be deleted from the user reject list of the owner.

For example:

Ron Underwood wants to delete dean.burrows@umts.uk from his user reject list (HTTP DIGEST authentication)

```
GET http://glms.operator.com/script?action=delete_user_reject_list_entry&owner=sip%3Aronald.underwood%40umts.uk&buddy_uri=sip%3Adean.burrows%40umts.uk
HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
  uri="http://glms.operator.com/script?action=delete_user_reject_list_entry&owner=sip%3Aronald.underwood%40umts.uk&buddy_uri=sip%3Adean.burrows%40umts.uk",
  response="e966c932a9242554e42c8ee200cec7f6",
  cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
  opaque="5ccc069c403ebaf9f0171e9517f40e41",
  qop=auth-int,
  nc=00000001

HTTP/1.1 200 OK
```

### 7.3.3.7 Modify user/group attributes in user reject list

This method shall modify attributes of entries in the user reject list.

Parameter	Value	Purpose
Action	modify_user_reject_list_entry	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Buddy_uri		URI of user or group to be modified ("*" is not allowed here).
Buddy_displayname		(optional) New display name of user or group.
services		(optional) List of service names formatted as described in 5.2.1.

As the result of invocation of this method, the GLMS shall set attributes of the entry identified by buddy\_uri parameter in the user accept list to the values supplied in this request. If an optional parameter is not present in the request, then the corresponding attribute shall not be changed.

If the value of buddy\_displayname parameter is an empty string then the modified entry in the user reject list shall have no display name after the method is invoked.

If the services parameter is an empty string then the modified entry in the user reject list shall be valid for all services.

Note: This behavior allows for extensibility of the protocol and allows reuse of this method for changing various other attributes that might be added in future versions of the specification.

For example:

Ron Underwood wants to change display name of dean.burrows@umts.uk in the user reject list (HTTP Digest authentication) and wants dean.burrows@umts.uk to be in user reject list for PoC and presence

```
GET http://glms.operator.com/script?action=modify_user_reject_list_entry&owner=sip%3Aronald.underwood%40umts.uk&buddy_uri=sip%3Adean.burrows%40umts.uk&buddy_displayname=Supporter&services=poc+presence
HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
  uri="http://glms.operator.com/script?action=modify_user_reject_list_entry&owner=sip%3Aronald.underwood%40umts.uk&buddy_uri=sip%3Adean.burrows%40umts.uk&buddy_displayname=Supporter&services=poc+presence",
  response="e966c932a9242554e42c8ee200cec7f6",
  cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
  opaque="5ccc069c403ebaf9f0171e9517f40e41",
  qop=auth-int,
  nc=00000001

HTTP/1.1 200 OK
```

### 7.3.3.8 Retrieve user reject list

This method shall retrieve the contents of the user reject list.

Parameter	Value	Purpose
Action	get_user_reject_list	The type of action to be done.
Owner		A SIP URI of the user performing the action.

As the result of invocation of this method, the GLMS generates a document of content-type application/buddies+xml (see sub-clause 7.5.3) encoded as UTF-8 (see reference [11]) describing the contents of the user accept list and shall return it in the HTTP "200 OK" final response to the initiator of the request.

For example:

Ron Underwood wants to retrieve content of his reject list (HTTP DIGEST authentication)

```
GET http://glms.operator.com/script?action=get_user_reject_list&owner=sip%3Aronald.underwood%40umts.uk HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=get_user_reject_list&owner=sip%3Aronald.underwood%40umts.uk",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

Content-Type: application/buddies+xml; charset=UTF-8

```
<?xml version="1.0" encoding="UTF-8"?>
<buddies>
  <buddy uri="*" services="presence"/>
  <buddy displayname="Uncle" uri="sip:vernon.keel@umts.uk" services="presence"/>
  <buddy uri="sip:tom.nettle@umts.uk" services="poc presence"/>
</buddies>
```

### 7.3.3.9 Modify Access List Attributes

This method shall modify attributes common to both access list.

Parameter	Value	Purpose
Action	modify_access_lists	The type of action to be done.
Owner		A SIP URI of the user performing the action.
in_use	"Y"   "N"	(optional) A value defining whether access lists are "active".

As the result of invocation of this method, the GLMS shall set common access list attributes to their new values as indicated by the method parameters. If an optional parameter is not present in the request, then the value of the corresponding attribute shall not be changed. If no optional attribute is present, then the GLMS shall generate a document of content-type "application/access-lists+xml" (see sub-clause 7.5.4) encoded as UTF-8 (see reference [11]) describing the values of all common access list attributes and shall return it in the HTTP "200 OK" final response to the request.

Note: This behavior allows for extensibility of the protocol and allows reuse of this method for changing various other attributes that might be added in future versions of the specification.

For example:

Ron Underwood wants to temporarily deactivate his reject and accept lists (HTTP DIGEST authentication)

```
GET http://glms.operator.com/script?action=modify_access_lists&owner=sip%3Aronald.underwood%40umts.uk&in_use=N HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=modify_access_lists&owner=sip%3Aronald.underwood%40umts.uk",
```

```

response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001

```

HTTP/1.1 200 OK

Ron Underwood wants to query his access list attributes (HTTP Digest authentication)

```

GET http://glms.operator.com/script?action=modify_access_lists&owner=sip%3Aronal
d.underwood%40umts.uk HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=modify_access_lists&owner=sip%3
Aronald.underwood%40umts.uk",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001

```

HTTP/1.1 200 OK

Content-Type: application/access-lists+xml;charset=UTF-8

```

<?xml version="1.0" encoding="UTF-8"?>
<access_lists in_use="Y"/>

```

## 7.3.4 Groups

Im operations for Groups include

- Create/delete Group
- Add/delete/modify user identity
- Store/Retrieve Groups (storing is done implicitly by other operations)
- Modify Groups

### 7.3.4.1 Create Group

This method shall creating group. The GLMS shall generate a unique SIP URI for the group and shall return it to the UE in the success response to this request. The actual SIP URI generation mechanism is implementation defined.

Parameter	Value	Purpose
Action	create_group	The type of action to be done.
Owner		A SIP URI of the user performing the action.
group_displayname		Display name of the group being added.
Session_type	"chat"   "instant"	A value defining type of the talk session.
Membership	"open"   "restricted"	A value defining type of a group membership.
max_participants		(optional) Maximum number of participants of the talk session. If not present then value of zero is assumed. The number of participants shall be passed as a string representation of the decimal number. If a value of "max" is used, then the value of max_participants shall be the maximum allowed number set by the operator.
anonymous	"Y"   "N"	(optional) A value defining whether all the group users are anonymous.
restrict_init	"Y"   "N"	(optional) A value defining whether only the owner of the group is allowed to start the group talk.

restrict_add	"Y"   "N"	(optional) A value defining whether only the owner of the group is allowed to add participants to talk.
--------------	-----------	---

As the result of invocation of this method, the GLMS shall create a new group for the owner, shall generate a unique SIP URI for it, shall set the group attributes to the requested values and shall return in the HTTP "200 OK" final response a document of content type application/group+xml (see sub-clause 7.5.5) encoded as UTF-8 (see reference [11]) that describes the created group. The description shall contain the generated SIP URI of the group.

The newly created group shall have defined attributes (session\_type, membership, max\_participants) and empty group member list and empty group reject list.

The value of the group\_displayname must not be an empty string.

The value "Y" of the anonymous parameter indicates that the PoC server shall hide identities of all session participants. The value "N" of the anonymous parameter indicates that the PoC server shall not hide identity of any session participants. If the anonymous parameter is omitted the semantics is the same as if it was present and the value was "N".

The value "Y" of the restrict\_init parameter indicates that the PoC server shall allow only the owner of the group to start the group talk. The value "N" of the restrict\_init parameter indicates that the PoC server shall allow any user who is allowed to access the group talk according to group member list and group reject list to start the group talk. If the restrict\_init parameter is omitted the semantics is the same as if it was present and the value was "N".

The value "Y" of the restrict\_add parameter indicates that the PoC server shall allow only the owner of the group to add users to group talk. The value "N" of the restrict\_add parameter indicates that the PoC server shall allow any participant of the group talk to request adding users to the group talk. If the restrict\_add parameter is omitted the semantics is the same as if it was present and the value was "N".

**Note:** If the user sets the max\_participants attribute to value "max" the GLMS shall not store the current maximum allowed value. Instead an information that the user requested the "maximum allowed" number shall be stored. This way it shall be ensured that the max\_participants is always set to the maximum operator allowed number if requested by the user. **Note:** The anonymous parameter is only valid for groups of type "chat". Nevertheless, the GLMS shall store the parameter value as specified in the request.

For example, the URI can be generated based on the SIP URI of the owner and the display name of the group as shown in the following example. Various other ways of generating the SIP URI are possible. In the example below, the presence of "group" tag in the SIP URI is not required when using other SIP URI generation mechanisms.

For example:

Ron Underwood wants to create a group named "Group 01" (HTTP DIGEST authentication)

```
GET http://glms.operator.com/script?action=create_group&owner=sip%3Aronald.underwood%40umts.uk&group_displayname=Group+01&session_type=chat&membership=restricted&max_participants=7&restrict_init=Y HTTP/1.1
```

```
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=create_group&owner=sip%3Aronald.underwood%40umts.uk&group_displayname=Group+01&session_type=chat&membership=restricted&max_participants=7",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

```
HTTP/1.1 200 OK
```

```
Content-Type: application/group+xml;charset=UTF-8
```

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<group uri="sip:ronald.underwood@umts.uk;group=Group_01"/>
```

The HTTP "200 OK" response may also contain other XML attributes according to the XML schema. These attributes shall not be included, if the GLMS has accepted the values specified by the user in the GET request. If the server

overrides some of these values, it shall add the respective XML element to the HTTP “200 OK” response that informs the initiator of the request that the attribute was set to another value than requested. For example when the participant requests the attribute “max\_participants” to have value “max” (unlimited number of participant) and the server is configured not to allow this and the maximum allowed number is twenty, the call flow may look like this.

```
GET http://glms.operator.com/script?action=create_group&owner=sip%3Aronald.underwood%40umts.uk&group_displayname=Group+01&session_type=chat&membership=restricted&max_participants=max HTTP/1.1
```

```
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=create_group&owner=sip%3Aronald.underwood%40umts.uk&group_displayname=Group+01&session_type=chat&membership=restricted&max_participants=max",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

```
HTTP/1.1 200 OK
Content-Type: application/group+xml;charset=UTF-8
```

```
<?xml version="1.0" encoding="UTF-8"?>
<group uri="sip:a234bob@poc345.umts.uk" max_participants="20"/>
```

### 7.3.4.2 Delete Group

This method shall delete an existing group.

Parameter	Value	Purpose
Action	delete_group	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Group_uri		URI of the group to be deleted.

As the result of invocation of this method, the group identified by group\_uri shall be deleted including all entries on the group member list and group reject list.

For example:

Ron Underwood wants to delete a group named "Group\_01" (HTTP Digest authentication)

```
GET http://glms.operator.com/script?action=delete_group&owner=sip%3Aronald.underwood@umts.uk&group_uri=sip%3Aa234bob%40poc345.umts.uk HTTP/1.1
```

```
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=delete_group&owner=sip%3Aronald.underwood@umts.uk&group_uri=sip%3Aa234bob%40poc345.umts.uk",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

```
HTTP/1.1 200 OK
```

### 7.3.4.3 Add user identity to group member list

This method shall insert a new user identity to group member list.

Parameter	Value	Purpose
Action	add_member_list_entry	The type of action to be done.
Owner		A SIP URI of the user performing the action.
Group_uri		URI of the group to be modified.
Buddy_uri		URI of user to be added.

Buddy\_displayname | (optional) Display name of user being added.

As the result of invocation of this method, the user identified by the buddy\_uri parameter with display name in buddy\_displayname shall be inserted to the member list of the group identified by group\_uri parameter.

If the buddy\_displayname parameter is not present or its value is an empty string then the new entry in the contact list shall have no display name.

Note: The type of the inserted user must not be “contact list” or “group”, it must be “user”. The parameter specifying the type is therefore omitted.

For example:

Ron Underwood wants to add jim.gilder@umts.uk into group member list of his group previously created as shown in section 7.3.4.1 (HTTP Digest authentication)

```
GET http://glms.operator.com/script?action=add_member_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aa234bob%40poc345.umts.uk&buddy_uri=sip%3Ajim.gilder@umts.uk&display=Jim HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=add_member_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aa234bob%40poc345.umts.uk&buddy_uri=sip%3Ajim.gilder@umts.uk&display=Jim",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

#### 7.3.4.4 Delete user identity from group member list

This method shall delete user identity from group member list.

Parameter	Value	Purpose
Action	delete_member_list_entry	The type of action to be done.
Owner		A SIP URI of the user performing the action.
group_uri		URI of the group to be modified.
buddy_uri		URI of user or contact list to be deleted.

As the result of invocation of this method, the user identity identified by buddy\_uri shall be deleted from the group member list of an existing group identified by group\_uri. The user identity being deleted must already exist in the group member list of the group.

For example:

Ron Underwood wants to remove vernon.keel@umts.uk from the group member list of his group previously created as shown in section 7.3.4.1 (HTTP DIGEST authentication)

```
GET http://glms.operator.com/script?action=delete_member_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aronald.underwood%40umts.uk%3Bgroup%3DGroup_01&buddy_uri=sip%3Avernon.keel%40umts.uk HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=delete_member_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aronald.underwood%40umts.uk%3Bgroup%3DGroup_01&buddy_uri=sip%3Avernon.keel%40umts.uk",
response="e966c932a9242554e42c8ee200cec7f6",
```

```

cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001

```

HTTP/1.1 200 OK

### 7.3.4.5 Modify user attributes in group member list

This method shall modify attributes of an entry in the group member list.

Parameter	Value	Purpose
Action	modify_member_list_entry	The type of action to be done.
Owner		A SIP URI of the user performing the action.
group_uri		URI of the group to be modified.
buddy_uri		URI of user to be modified.
buddy_displayname		(optional) New display name of user.

As the result of invocation of this method, the requested attributes of an existing entry identified by buddy\_uri in the group member list of an existing group identified by group\_uri shall be set to requested values. If an optional parameter is not present in the request, then the value of the corresponding attribute shall not be changed.

If the value of buddy\_displayname parameter is an empty string then the modified entry in the group member list shall have no display name after the method is invoked.

Note: This behavior allows for extensibility of the protocol and allows reuse of this method for changing various other attributes that might be added in future versions of the specification.

For example:

Ron Underwood wants to change display name of vernon.keel@umts.uk in the member list of his group previously created as shown in section 7.3.4.1 (HTTP DIGEST authentication)

```

GET http://glms.operator.com/script?action=modify_member_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aronald.underwood%40umts.uk%3Bgroup%3DGroup_01&buddy_uri=sip%3Avernon.keel%40umts.uk&buddy_displayname=Uncle+Vernon HTTP/1.1

```

```

Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=modify_member_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aronald.underwood%40umts.uk%3Bgroup%3DGroup_01&buddy_uri=sip%3Avernon.keel%40umts.uk&buddy_displayname=Uncle+Vernon",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001

```

HTTP/1.1 200 OK

### 7.3.4.6 Retrieve group member list

This method shall return the contents of the group member list.

Parameter	Value	Purpose
Action	get_member_list	The type of action to be done.
Owner		A SIP URI of the user performing the action.
group_uri		URI of the group whose members are to be retrieved.

As the result of invocation of this method, the GLMS shall generate a document of content type application/buddies+xml (see section 7.5.3) encoded as UTF-8 (see reference [11]) describing the contents of the group member list and shall return it in HTTP "200 OK" final response to the request.

The optional attribute “services” shall not be present in the response.

For example:

Ron Underwood wants to retrieve all group member list entries of his group previously created as shown in section 7.3.4.1 (HTTP Digest authentication)

```
GET http://glms.operator.com/script?action=get_member_list&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aa234bob%40poc345.umts.uk HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=get_member_list&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aa234bob%40poc345.umts.uk",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

Content-Type: application/buddies+xml;charset=UTF-8

```
<?xml version="1.0" encoding="UTF-8"?>
<buddies>
  <buddy displayname="Jim" uri="sip:jim.gilder@umts.uk"/>
  <buddy uri="sip:vernon.keel@umts.uk"/>
</buddies>
```

### 7.3.4.7 Add user list identity to group reject list

This method shall insert a new user identity to group reject list.

Parameter	Value	Purpose
Action	add_reject_list_entry	The type of action to be done.
Owner		A SIP URI of the user performing the action.
group_uri		URI of the group to be modified.
buddy_uri		URI of user to be added.
buddy_displayname		(optional) Display name of user or contact list being added.

As the result of invocation of this method, the user identified by the buddy\_uri parameter with display name in buddy\_displayname shall be inserted to the reject list of the group identified by group\_uri parameter.

If the buddy\_displayname parameter is not present or its value is an empty string then the new entry in the group reject list shall have no display name.

Note: The type of the inserted user must not be “contact list” or “group”, it must be “user”. The parameter specifying the type is therefore omitted.

For example:

Ron Underwood wants to add vernon.keel@umts.uk into group reject list of his group previously created as shown in section 7.3.4.1 (HTTP DIGEST authentication)

```
GET http://glms.operator.com/script?action=add_reject_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aronald.underwood%40umts.uk%3Bgroup%3DGroup_01&buddy_uri=sip%3Avernon.keel%40umts.uk HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=add_reject_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aronald.underwood%40umts.uk%3Bgroup%3DGroup_01&buddy_uri=sip%3Avernon.keel%40umts.uk",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
```

```
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

Ron Underwood wants to add agnus.singleton@umts.uk into group reject list of his group (HTTP Digest authentication)

```
GET http://glms.operator.com/script?action=add_reject_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aa234bob%40poc345.umts.uk&buddy_uri=sip%3Aagnus.singleton%40umts.uk&display=Agnus HTTP/1.1
```

```
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=add_reject_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aa234bob%40poc345.umts.uk&buddy_uri=sip%3Aagnus.singleton%40umts.uk&display=Agnus",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

### 7.3.4.8 Delete user identity from group reject list

This method shall delete user identity from group reject list.

Parameter	Value	Purpose
action	delete_reject_list_entry	The type of action to be done.
owner		A SIP URI of the user performing the action.
group_uri		Uri of the group to be modified.
buddy_uri		URI of user or contact list to be deleted.

As the result of invocation of this method, the user identity identified by buddy\_uri shall be deleted from the group reject list of an existing group identified by group\_uri. The user identity being deleted must already exist in the group reject list of the group.

For example:

Ron Underwood wants to remove agnus.singleton@umts.uk from the group reject list of his group previously created as shown in section 7.3.4.1 (HTTP Digest authentication)

```
GET http://glms.operator.com/script?action=delete_reject_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aa234bob%40poc345.umts.uk&buddy_uri=sip%3Aagnus.singleton%3Bumts.uk HTTP/1.1
```

```
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=delete_reject_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aa234bob%40poc345.umts.uk&buddy_uri=sip%3Aagnus.singleton%3Bumts.uk",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

### 7.3.4.9 Modify user attributes in group reject list

This method shall modify attributes of an entry in the group reject list.

Parameter	Value	Purpose
-----------	-------	---------

action	modify_reject_list_entry	The type of action to be done.
owner		A SIP URI of the user performing the action.
group_uri		URI of the group to be modified.
buddy_uri		URI of user to be modified.
buddy_displayname		(optional) New display name of user.

As the result of invocation of this method, the requested attributes of an existing entry identified by `buddy_uri` in the group reject list of an existing group identified by `group_uri` shall be set to requested values. If an optional parameter is not present in the request, then the value of the corresponding attribute shall not be changed.

If the value of `buddy_displayname` parameter is an empty string then the modified entry in the group reject list shall have no display name after the method is invoked.

Note: This behavior allows for extensibility of the protocol and allows reuse of this method for changing various other attributes that might be added in future versions of the specification.

For example:

Ron Underwood wants to change display name of `agnus.singleton@umts.uk` in the group reject list of his group previously created as shown in section 7.3.4.1 (HTTP DIGEST authentication)

```
GET http://glms.operator.com/script?action=modify_reject_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aronald.underwood%40umts.uk%3Bgroup%3DGroup_01&buddy_uri=sip%3Aagnus.singleton%40umts.uk&buddy_displayname=Accuser HTTP/1.1
```

```
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=modify_reject_list_entry&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aronald.underwood%40umts.uk%3Bgroup%3DGroup_01&buddy_uri=sip%3Aagnus.singleton%40umts.uk&buddy_displayname=Accuser",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

### 7.3.4.10 Retrieve group reject list

This method shall return the contents of the group member list.

Parameter	Value	Purpose
action	get_reject_list	The type of action to be done.
owner		A SIP URI of the user performing the action.
group_uri		URI of the group to be retrieved.

As the result of invocation of this method, the GLMS shall generate a document of content type `application/buddies+xml` (see section 7.5.3) encoded as UTF-8 (see reference [11]) describing the contents of the group reject list and shall return it in HTTP “200 OK” final response to the request.

The optional attribute “services” shall not be present in the response.

For example:

Ron Underwood wants to retrieve all group reject list entries of his group previously created as shown in section 7.3.4.1 (HTTP DIGEST authentication)

```
GET http://glms.operator.com/script?action=get_reject_list&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aronald.underwood%40umts.uk%3Bgroup%3DGroup_01 HTTP/1.1
```

```
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=get_reject_list&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aronald.underwood%40umts.uk%3Bgroup%3DGroup_01",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

Content-Type: application/buddies+xml;charset=UTF-8

```
<?xml version="1.0" encoding="UTF-8"?>
<buddies>
  <buddy displayname="Agnus" uri="sip:agnus.singleton@umts.uk"/>
</buddies>
```

### 7.3.4.11 Get list of groups

This method shall retrieve a list of all groups belonging to the subscriber (owner).

Parameter	Value	Purpose
Action	get_list_of_groups	The type of action to be done.
Owner		A SIP URI of the user performing the action.

As the result of invocation of this method, the GLMS shall return in HTTP “200 OK” final response a document of content type “application/groups+xml” (see sub-clause 7.5.6) encoded as UTF-8 (see reference [11]) describing all the groups that belong to the invoking subscriber.

For example:

Ron Underwood wants to retrieve a list of all his groups (HTTP DIGEST authentication)

```
GET http://glms.operator.com/script?action=get_list_of_groups&owner=sip%3Aronald.underwood%40umts.uk HTTP/1.1
```

```
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="http://glms.operator.com/script?action=get_list_of_groups&owner=sip%3Aronald.underwood%40umts.uk",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

HTTP/1.1 200 OK

Content-Type: application/groups+xml;charset=UTF-8

```
<?xml version="1.0" encoding="UTF-8"?>
<groups>
  <group displayname="Group" uri="sip:ronald.underwood@umts.uk;group=Group_01"
    session_type="chat" membership="restricted" max_participants="7"/>
</groups>
```

### 7.3.4.12 Modify group attributes

This method shall modify group attributes.

Parameter	Value	Purpose
Action	set_group_attributes	The type of action to be done.
Owner		A SIP URI of the user performing the action.

group_uri		URI of the group to be modified.
group_displayname		(optional) New display name of the group.
Session_type	"chat"   "instant"	(optional) A value defining type of the talk session.
membership	"open"   "restricted"	(optional) A value defining type of a group membership.
max_participants		(optional) Maximum number of participants of the session. If not present then value the attribute shall not be changed. The number of participants shall be passed as a string representation of the decimal number. If a value of "max" is used, then the value of max_participants shall be the maximum allowed number set by the operator.
anonymous	"Y"   "N"	(optional) A value defining whether all the group users are anonymous.
restrict_init	"Y"   "N"	(optional) A value defining whether only the owner of the group is allowed to start the group talk.
restrict_add	"Y"   "N"	(optional) A value defining whether only the owner of the group is allowed to add participants to talk.

As the result of invocation of this method, the GLMS shall set the attributes of an existing group identified by group\_uri to their new values as request in the method parameters. If an optional parameter is not present in the request, then the value of the corresponding attribute shall not be changed. If attribute value update executed successfully the GLMS shall respond to the request with an HTTP "200 OK" final response.

The value of group\_displayname parameter must not be an empty string.

The value "Y" of the anonymous parameter indicates that the PoC server shall hide identities of all session participants. The value "N" of the anonymous parameter indicates that the PoC server shall not hide identity of any session participants. If the anonymous parameter is omitted the value of the anonymous attribute shall not be changed.

The value "Y" of the restrict\_init parameter indicates that the PoC server shall allow only the owner of the group to start the group talk. The value "N" of the restrict\_init parameter indicates that the PoC server shall allow any user who is allowed to access the group talk according to group member list and group reject list to start the group talk. If the restrict\_init parameter is omitted the semantics is the same as if it was present and the value was "N".

The value "Y" of the restrict\_add parameter indicates that the PoC server shall allow only the owner of the group to add users to group talk. The value "N" of the restrict\_add parameter indicates that the PoC server shall allow any participant of the group talk to request adding users to the group talk. If the restrict\_add parameter is omitted the semantics is the same as if it was present and the value was "N".

Note: This behavior allows for extensibility of the protocol and allows reuse of this method for changing various other attributes that might be added in future versions of the specification.

Note: If the user sets the max\_participants attribute to value "max" the GLMS shall not store the current maximum allowed value. Instead an information that the user requested the "maximum allowed" number shall be stored. This way it shall be ensured that the max\_participants is always set to the maximum operator allowed number if requested by the user. Note: The anonymous parameter is only valid for groups of type "chat". Nevertheless, the GLMS shall store the parameter value as specified in the request.

As in chapter 7.3.4.1 in case that the parameters specified by the user were overridden by the server the HTTP "200 OK" response to this request shall contain body of type "application/group+xml" (see sub-clause 7.5.5) encoded as UTF-8 (see reference [11]) describing the values of the overridden parameters.

For example:

Ron Underwood wants to change display name of the group previously created as shown in section 7.3.4.1 (HTTP digest authentication)

```
GET http://glms.operator.com/script?action=set_group_attributes&owner=sip%3Aronald.underwood%40umts.uk&group_uri=sip%3Aronald.underwood%40umts.uk%3Bgroup%3DGroup_01&group_displayname=Friends HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
```

```
uri="http://glms.operator.com/script?action=set_group_attributes&owner=sip%
3Aronald.underwood%40umts.uk&group_uri=sip%3Aronald.underwood%40umts.uk%3Bg
roup%3DGroup_01&group_displayname=Friends",
response="e966c932a9242554e42c8ee200cec7f6",
cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
qop=auth-int,
nc=00000001
```

```
HTTP/1.1 200 OK
Content-Type: application/group+xml;charset=UTF-8
```

```
<?xml version="1.0" encoding="UTF-8"?>
<group displayname="Friends" uri="sip:ronald.underwood@umts.uk;group=Group_01"
  session_type="chat" membership="restricted" max_participants="7"/>
```

Ron Underwood wants to set the group to have open membership (HTTP Digest authentication)

```
GET http://glms.operator.com/script?action=set_group_attributes&owner=sip%3Arona
ld.underwood%40umts.uk&group_uri=sip%3Aa234bob%40poc345.umts.uk&membership=
open HTTP/1.1
Authorization: Digest username="u29502566", realm="glms.umts.uk",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
  uri="http://glms.operator.com/script?action=set_group_attributes&owner=sip%
3Aronald.underwood%40umts.uk&group_uri=sip%3Aa234bob%40poc345.umts.uk&membe
rship=open",
  response="e966c932a9242554e42c8ee200cec7f6",
  cnonce="dcd99agsfgfsa8b7102dd2f0e8b1",
  opaque="5ccc069c403ebaf9f0171e9517f40e41",
  qop=auth-int,
  nc=00000001
```

```
HTTP/1.1 200 OK
Content-Type: application/group+xml;charset=UTF-8
```

```
<?xml version="1.0" encoding="UTF-8"?>
<group displayname="Group" uri="sip:ronald.underwood@umts.uk;group=Group_01"
  session_type="chat" membership="open" max_participants="7"/>
```

### 7.3.5 Caching support

The caching uses a mechanism defined in RFC2616 [7].

The GLMS server shall hold a timestamp with each document stored on the server (contact list, list of contact lists, etc.). The timestamp shall be generated when the document is created and updated with each change of the document. When the server responds to a request with a HTTP “200 OK” response it shall include the “Last-Modified” header to the response.

The UE may use the timestamp to perform a “conditional GET” request. In case that the UE has some version of the required document, it may include the timestamp information in the “If-Modified-Since” in the GET request. If the document was not modified since the specified time then the GLMS server responds with a HTTP “304 Not Modified” response. In the other case the GLMS server responds with a HTTP “200 OK” response, same as in the previous paragraph.

As described above, the “Last-Modified” header is present in all HTTP “200 OK” responses including those to request for modification of server stored data.

If the UE has a locally available version of the document it is going to modify, it may include the “If-Unmodified-Since” header in the request. The GLMS shall verify that the timestamp passed in the “If-Unmodified-Since” header is the same as the timestamp of the document stored on the server. In such case the action is first performed, the document on the GLMS is updated and the new timestamp of the document is returned in the HTTP “Last-Modified” header in the HTTP “200 OK” final response. The UE shall update its local copy of the modified document and shall assign the

timestamp returned from the GLMS. With using the “If-Unmodified-Since” header the GLMS and the UE are able to produce a coherent state after the operation.

For example:

A UE application that has a persistent storage available downloads a document when it starts the first time.

```
GET http://glms.operator.com?somerequest HTTP/1.1

HTTP/1.1 200 OK
Date: Fri, 18 Oct 2002 12:00:41 GMT
Last-Modified: Fri, 18 Oct 2002 09:10:00 GMT
Content-Type: ...

<document>
```

After fetching the document, the UE places it into the cache in the persistent storage together with its timestamp.

The next time the UE application inspects its cache for a version of the required document and realizes that the document is present with timestamp “Fri, 18 Oct 2002 09:10:00 GMT”. Therefore, it issues a conditional GET request:

```
GET http://glms.operator.com?somerequest HTTP/1.1
If-Modified-Since: Fri, 18 Oct 2002 9:10:00 GMT

HTTP/1.1 304 Not Modified
Date: Fri, 18 Oct 2002 12:30:13 GMT
Last-Modified: Fri, 18 Oct 2002 09:10:00 GMT
```

When the UE application wants to change the document downloaded at the beginning of this example, it uses the conditional GET to perform the operation.

```
GET http://glms.operator.com?modifyrequest HTTP/1.1
If-Unmodified-Since: Fri,18 Oct 2002 9:10:00 GMT

HTTP/1.1 200 OK
Date: Fri, 18 Oct 2002 12:59:38 GMT
Last-Modified: Fri, 18 Oct 2002 12:59:40 GMT
```

After the GLMS confirms the success of the operation requested, the UE application performs the same operation on the locally stored data to produce a state coherent with the server and will assign the delivered timestamp to the newly produced document.

## 7.4 Error cases

### 7.4.1 Conflicts

The protocol defined in this document allows for conflicts to arise. These conflict result from that not all of the combination of attributes are allowed. In such cases the GLMS server shall respond to the request with HTTP “409 Conflict” response. An example of a conflict is when a user sets type attribute of group being (open,chat) to “instant” since the combination (instant,chat) is not allowed in the PoC service.

In situation when a user wants to add an entry to a list that is already present in the list the GLMS shall assure that the result of the operation is the same as if the entry was not present in the list before.

In situation when a user wants to set an attribute (including “in use” and “Do-not-Disturb”) to the same value as it is set currently, then the GLMS shall not change the value and shall respond with HTTP “200 OK” response.

According to chapter 5.2 the GLMS shall check that the set of services for which an entry is valid in user reject list and the set of services for which the entry (with the same URI) is valid in user accept list are disjoint. If the user requests an

operation to be performed that breaks this rule the GLMS shall not allow this action and shall respond with HTTP “409 Conflict” response.

The GLMS should provide additional information about the conflict reason in clear text in the reason phrase of the HTTP 409 response, e.g. “409 Combination instant/chat is not allowed”. The UE may display this information to the user to inform him about the reason for which the requested operation was rejected.

## 7.4.2 Status codes and descriptions

The result of the action is indicated by status code of the HTTP final response.

Following error codes shall be used:

Status code and reason phrase	Description of meaning
200 OK	The action finished OK
304 Not Modified	The requested document has not been modified (see 7.3.5)
400 Bad Request	The request could not be understood by the server due to malformed syntax (missing required parameters, unknown action requested, broken HTTP syntax). The client should not repeat the request without modifications.
401 Unauthorized	There are problems with authorization
402 Payment Required	Action was not executed due to charging policy of the operator (for example low credit when using pre-paid charging)
403 Forbidden	The authenticated user is not allowed to perform the action (for example if someone tries to delete group that is owned by someone else).
404 Not Found	Requested list or attribute or entry does not exist (Also used for delete operations.)
405 Method Not Allowed	Not supported method was used in the request. Only GET method is supported.
407 Proxy Authentication Required	When using a proxy, the proxy may use this method to request authentication.
408 Request Timeout	The client did not produce a request within the time that the server was prepared to wait. The client may repeat the request without modifications at any later time.
409 Conflict	Conflict occurred.
412 Precondition Failed	Precondition given in “If-Unmodified-Since” header failed. The UE should update list by downloading before repeating to modify it.

While this specification suggests specific wording for reason phrases, implementations may choose other text, for example, in the language indicated in the Accept-Language header field of the request.

## 7.5 Protocol XML schemas

### 7.5.1 Content application/buddylist+xml

Content type „application/buddylist+xml“ is defined as an XML structure with the following XML Schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="buddylist" type="buddylist"/>

  <xs:complexType name="buddylist">
    <xs:attribute name="displayname" type="xs:string" use="optional"/>
    <xs:attribute name="uri" type="xs:anyURI" use="required"/>
    <xs:attribute name="default" type="yes_no" use="optional"/>
    <xs:attribute name="type" type="list_type" use="required"/>
  </xs:complexType>

  <xs:simpleType name="yes_no">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Y"/>
      <xs:enumeration value="N"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

```

</xs:simpleType>

<xs:simpleType name="list_type">
  <xs:restriction base="xs:string">
    <xs:enumeration value="user"/>
    <xs:enumeration value="group"/>
  </xs:restriction>
</xs:simpleType>

</xs:schema>

```

## 7.5.2 Content application/buddylists+xml

Content type „application/buddylists+xml“ is defined as an XML structure with the following XML Schema:

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="buddylists" type="buddylists"/>

  <xs:complexType name="buddylists">
    <xs:sequence>
      <xs:element name="buddylist" type="buddylist" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="buddylist">
    <xs:attribute name="displayname" type="xs:string" use="optional"/>
    <xs:attribute name="uri" type="xs:anyURI" use="required"/>
    <xs:attribute name="default" type="yes_no" use="optional"/>
    <xs:attribute name="type" type="list_type" use="required"/>
  </xs:complexType>

  <xs:simpleType name="yes_no">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Y"/>
      <xs:enumeration value="N"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="list_type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="user"/>
      <xs:enumeration value="contact list"/>
      <xs:enumeration value="group"/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>

```

## 7.5.3 Content application/buddies+xml

Content type „application/buddies+xml“ is defined as an XML structure with the following XML Schema:

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="buddies" type="buddies">

  <xs:complexType name="buddies">
    <xs:sequence>
      <xs:element name="buddy" type="buddy" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="buddy">
    <xs:attribute name="displayname" type="xs:string" use="optional"/>
    <xs:attribute name="uri" type="xs:anyURI" use="required"/>
    <xs:attribute name="services" type="xs:string" use="optional"/>
  </xs:complexType>

```

```

</xs:complexType>
</xs:schema>

```

## 7.5.4 Content application/access-lists+xml

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="access_lists" type="access_lists"/>
  <xs:complexType name="access_lists">
    <xs:attribute name="in_use" type="yes_no" use="required"/>
  </xs:complexType>

  <xs:simpleType name="yes_no">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Y"/>
      <xs:enumeration value="N"/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>

```

## 7.5.5 Content application/group+xml

Content type „application/group+xml“ is defined as an XML structure with the following XML Schema:

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="group" type="group"/>

  <xs:complexType name="group">
    <xs:attribute name="displayname" type="xs:string" use="optional"/>
    <xs:attribute name="uri" type="xs:anyURI" use="required"/>
    <xs:attribute name="session_type" type="session_type" use="optional"/>
    <xs:attribute name="membership" type="membership" use="optional"/>
    <xs:attribute name="max_participants" type="xs:integer"
      use="optional"/>
    <xs:attribute name="anonymous" type="yes_no" default="N"/>
    <xs:attribute name="restrict_init" type="yes_no" default="N"/>
    <xs:attribute name="restrict_add" type="yes_no" default="N"/>
  </xs:complexType>

  <xs:simpleType name="session_type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="chat"/>
      <xs:enumeration value="instant"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="membership">
    <xs:restriction base="xs:string">
      <xs:enumeration value="open"/>
      <xs:enumeration value="restricted"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="yes_no">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Y"/>
      <xs:enumeration value="N"/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>

```

## 7.5.6 Content application/groups+xml

Content type „application/groups+xml“ is defined as an XML structure with the following XML Schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="groups" type="groups"/>

  <xs:complexType name="groups">
    <xs:sequence>
      <xs:element name="group" type="group" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="group">
    <xs:attribute name="displayname" type="xs:string" use="optional"/>
    <xs:attribute name="uri" type="xs:anyURI" use="required"/>
    <xs:attribute name="session_type" type="session_type" use="required"/>
    <xs:attribute name="membership" type="membership" use="required"/>
    <xs:attribute name="max_participants" type="xs:integer"
      use="optional"/>
    <xs:attribute name="anonymous" type="yes_no" default="N"/>
  <xs:attribute name="restrict_init" type="yes_no" default="N"/>
  <xs:attribute name="restrict_add" type="yes_no" default="N"/>
  </xs:complexType>

  <xs:simpleType name="session_type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="chat"/>
      <xs:enumeration value="instant"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="membership">
    <xs:restriction base="xs:string">
      <xs:enumeration value="open"/>
      <xs:enumeration value="restricted"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="yes_no">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Y"/>
      <xs:enumeration value="N"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

## 7.5.7 Content application/attributes+xml

Content type „application/attributes+xml“ is defined as an XML structure with the following XML Schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="attributes" type="attributes"/>

  <xs:complexType name="attributes">
    <xs:sequence>
      <xs:element name="attribute" type="attribute" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="attribute">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="name" type="xs:string" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:schema>
```

```

    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

</xs:schema>

```

## 7.6 Protocol transport bindings

This specification follows the provisions in RFC2616 [7] with the following exceptions, clarifications and recommendations:

- the UE and GLMS shall support at least following transport bindings:
  - o HTTP 1.1
- the UE and the GLMS may support other bindings:
  - o HTTPS

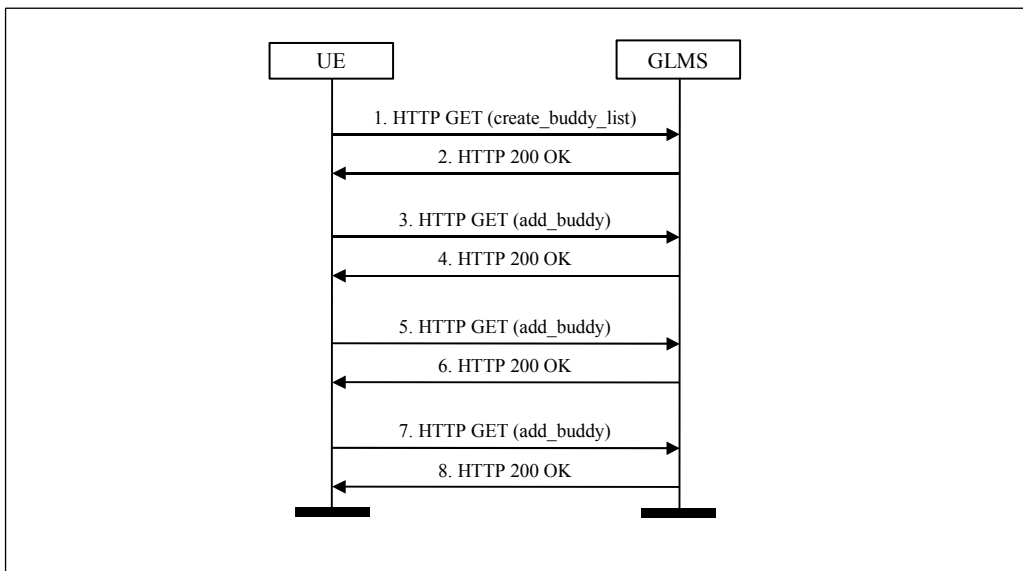
## 7.7 Typical UE to GLM procedures

This section is not normative. It shows several examples of usage of the protocol defined in this specification on the Im interface.

### 7.7.1 Contact list management

#### 7.7.1.1 Creating new contact list

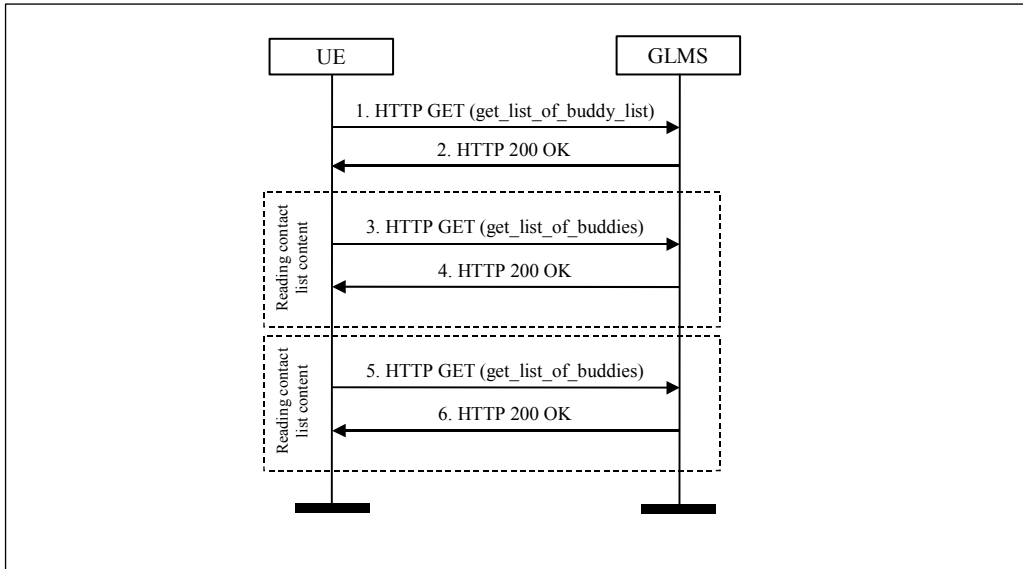
The user wants to create a new contact list that contains 3 members.



The user sends an HTTP GET request to create the new contact list (1-2). After the list is created, the user sends three HTTP GET requests to add three persons to this contact list (3-8).

#### 7.7.1.2 Reading contact lists

The user wants to obtain all his contact lists.

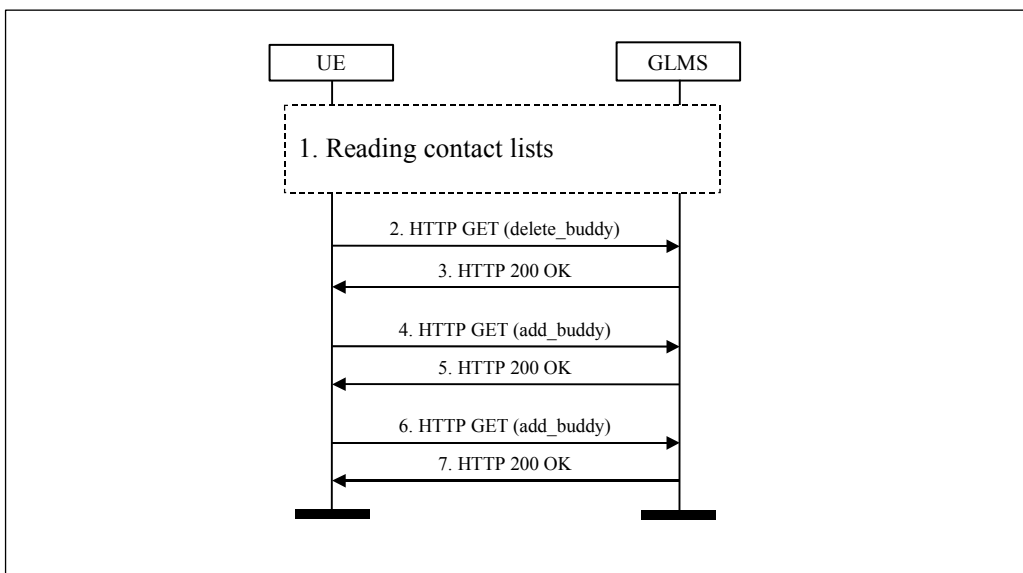


The user sends an HTTP GET request to obtain a list of all his contact lists (1-2). Then the user obtains a list of members for each contact list.

Let's assume the user has two contact lists. The user thus sends two HTTP GET requests (one request per contact list) to obtain list of contact list members (3-6).

### 7.7.1.3 Modifying an existing contact list

The user wants for example to remove one member of an existing contact list and add two new members to that list.

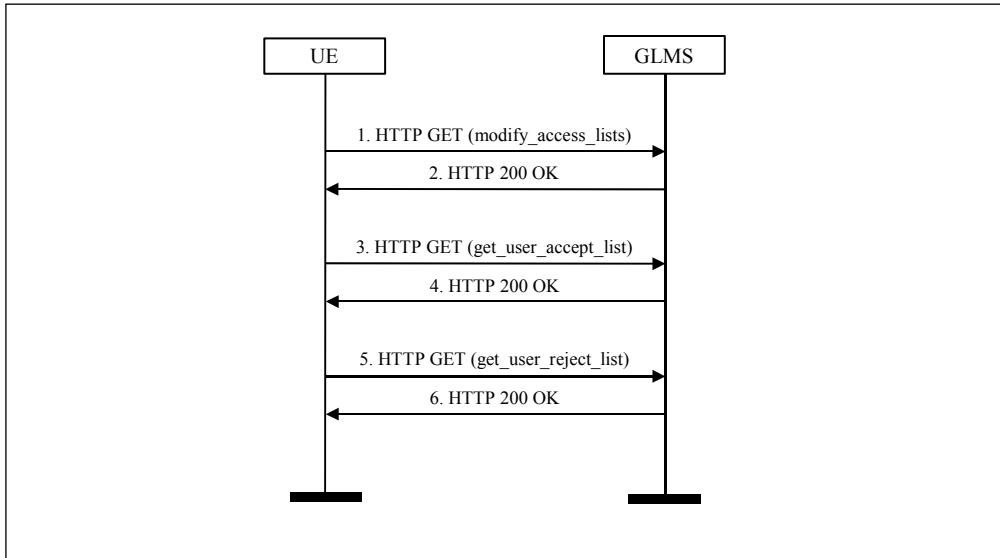


Prior modifying the user needs to know current list of existing contact lists (1). Then the user sends one HTTP GET request to delete an existing member (2-3), and two HTTP GET requests to add new list members (4-7).

## 7.7.2 Access list management

### 7.7.2.1 Reading access lists

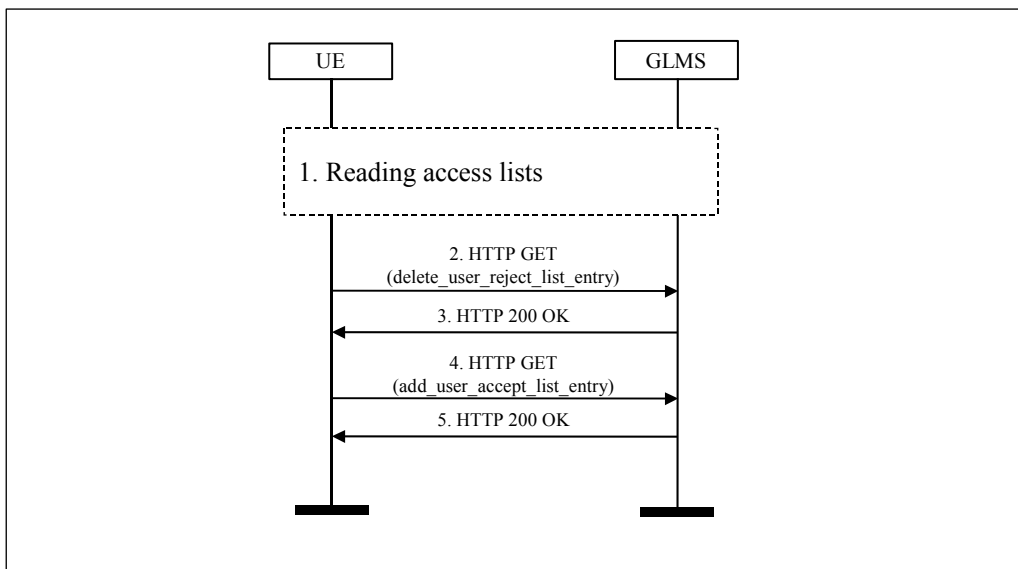
The user wants to obtain content of his access lists.



The user sends three HTTP GET requests: one request to obtain the value of the “in\_use” flag (1-2), second request to obtain the user accept list (3-4), third request to obtain the user reject list (5-6).

### 7.7.2.2 Modifying access lists

The user wants for example to remove one member from his user reject list and add this member into the user accept list.

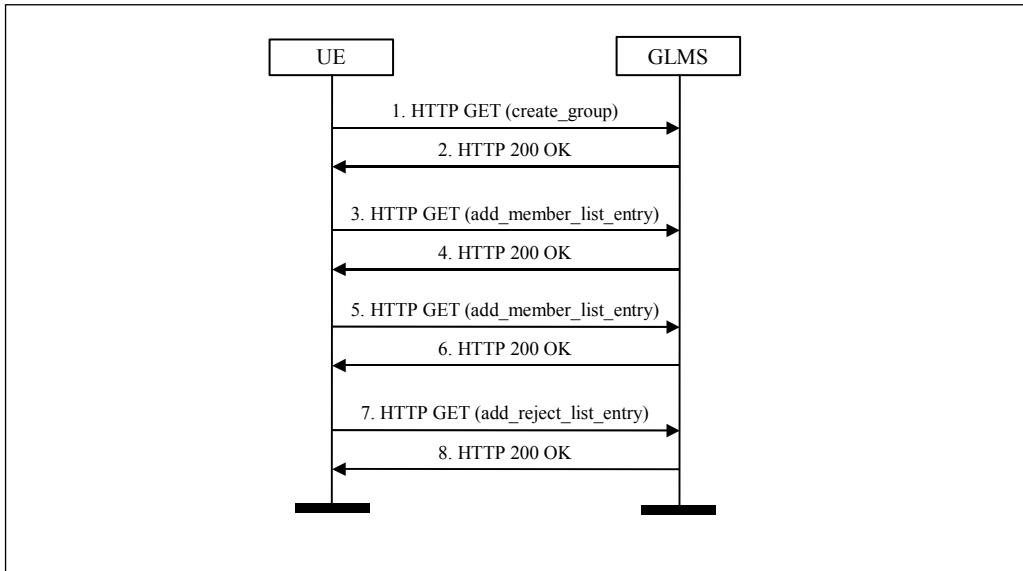


First the user needs to obtain content of access lists (1). Then the user sends one HTTP GET request to delete an user reject list member (2-3), and second HTTP GET request to insert new member into the user accept list (4-5).

## 7.7.3 Group list management

### 7.7.3.1 Creating new group

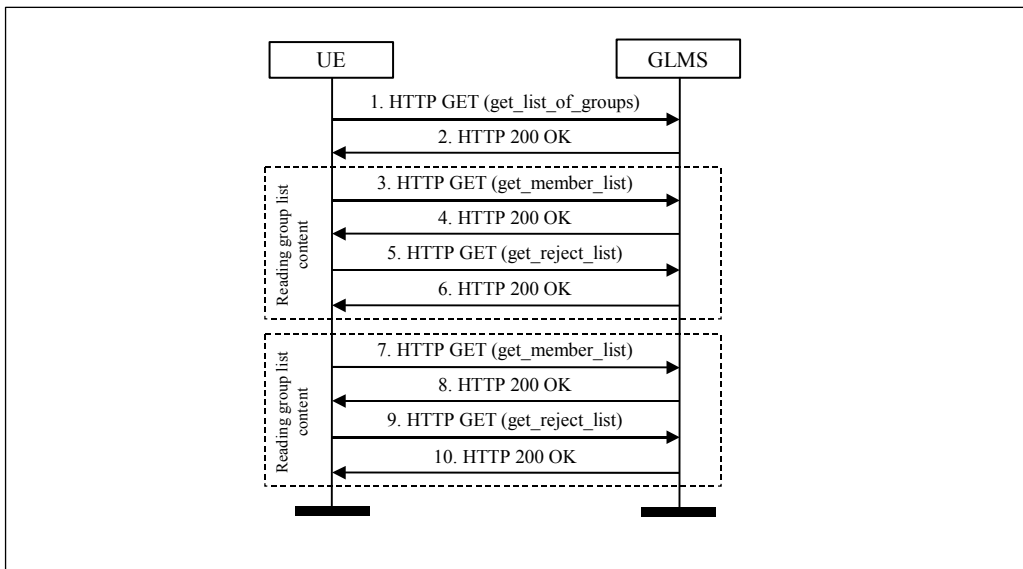
The user wants to create a new group. Group member list of the created group will contain 2 members; group reject list will have one member.



The user sends an HTTP GET request to create the new group (1-2). After the group is created, the user sends three HTTP GET requests: two requests to add items into member list (3-6), one request to add item into reject list (7-8).

### 7.7.3.2 Reading groups

The user wants to obtain all his groups.

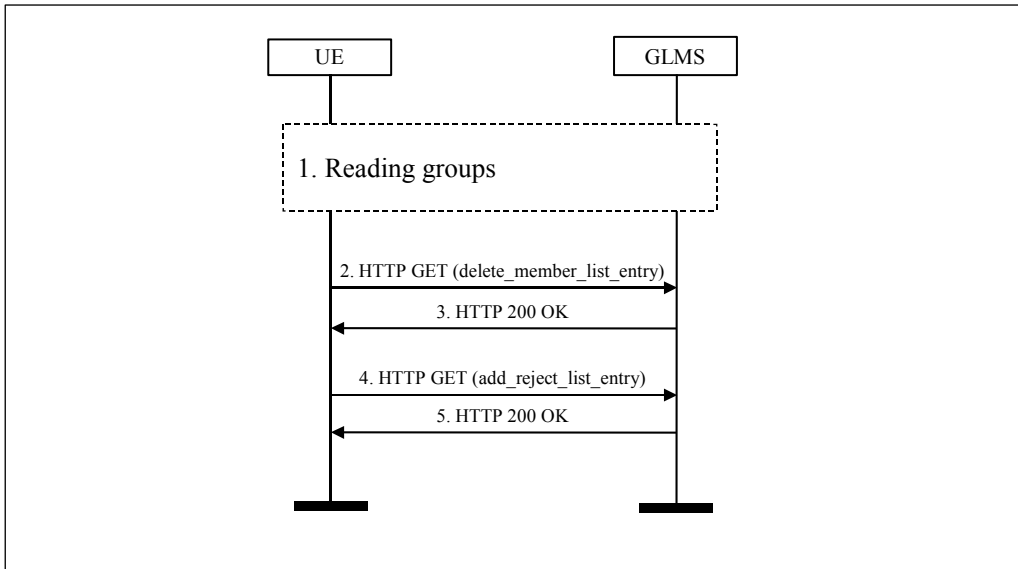


The user sends an HTTP GET request to obtain a list of all his groups (1-2). Then the user reads content of group member list and group reject list belonging to each group.

Let's assume the user has two groups. The user thus sends four HTTP GET requests (two requests per group) to obtain group member list and group reject lists contents (3-10).

### 7.7.3.3 Modifying group lists

The user wants for example to remove one subscriber out of the group member list and add this subscriber into the group reject list.

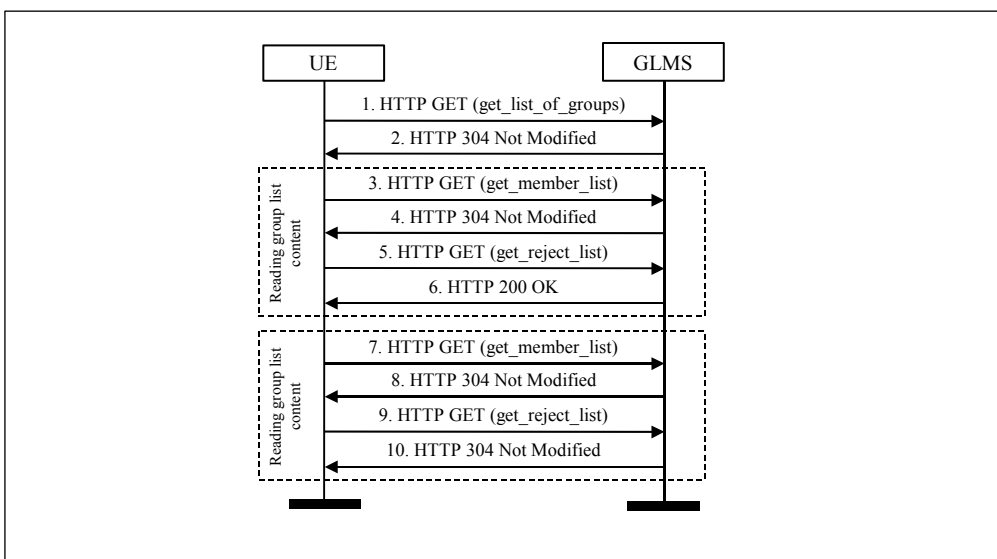


First, the user needs to obtain content of group lists (1). Then the user sends one HTTP GET request to delete a group member list item (2-3), and second HTTP GET request to insert new item into the group reject list (4-5).

## 7.7.4 Caching mechanism

### 7.7.4.1 Verifying locally stored data

The user wants to obtain all his groups. The UE has some version of the requested documents stored from previous interactions with the GLMS.



The UE performs several conditional GET operations to update the locally stored. The GET operations include the “If-Modified-Since” header containing the timestamp of the locally available document.

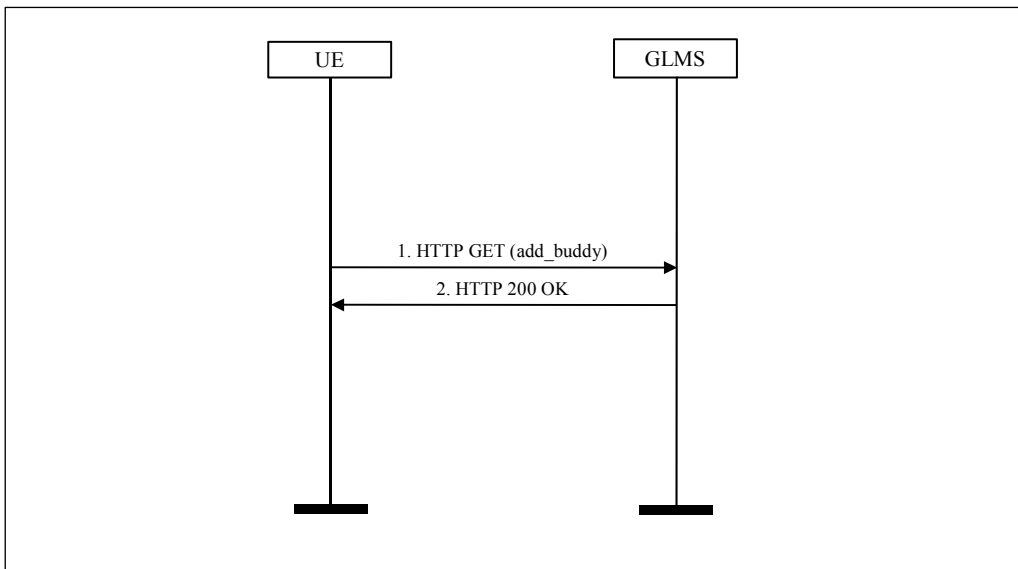
The user sends an HTTP GET request to update a list of all his groups (1-2). Since the local version of the document is up to date with the server version, the GLMS responds with HTTP “304 Not Modified”.

Then the user reads content of group member list and group reject list belonging to each group.

Let’s assume the user has two groups. The user thus sends four HTTP GET requests (two requests per group) to obtain group member list and group reject lists contents (3-10). The group reject list of the first group is not up to date. The GLMS responds with HTTP “200 OK” response to this request and returns the group reject list of the first group. Other locally stored data is up to date. The GLMS thus responds to the other request with HTTP “304 Not Modified” response.

#### 7.7.4.2 Modifying data

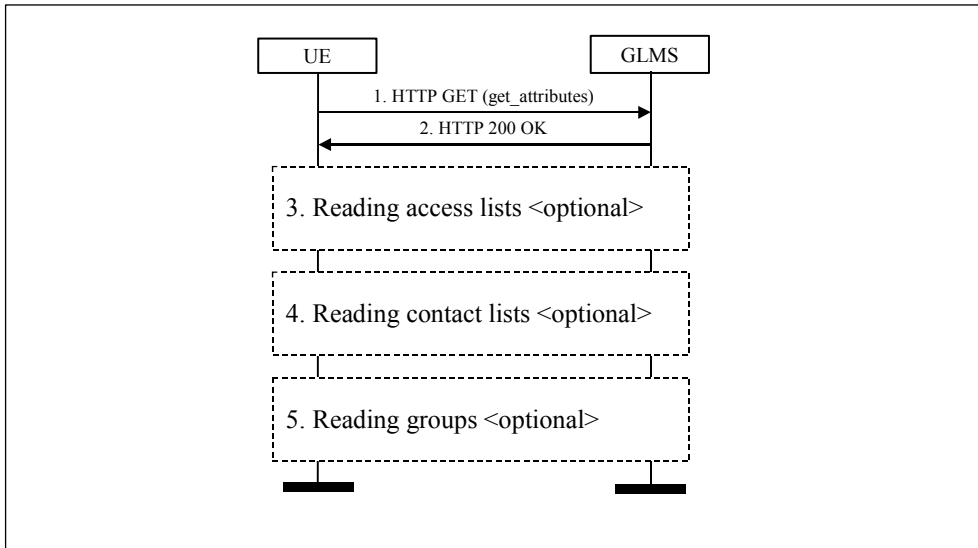
The user wants to add a buddy to one of his contact lists. Some version of the document describing the contact list is available locally.



To make sure that the remote version of the document stored on the GLMS is the same as the version of the document stored locally on the UE, the UE includes in the HTTP GET request (1) an “If-Unmodified-Since” header containing the timestamp of the locally stored document. Since the timestamp delivered in the request is the same as the timestamp of the document on the server, the server performs the action requested and responds with a HTTP “200 OK” response. The response contains the “Last-Modified” header with the new timestamp of the document produced by the action requested.

#### 7.7.5 Start-up procedure

The user switches on his UE. The PoC application is started automatically, registers to the IMS core and downloads the required data from the GLMS. The download of the data from GLMS shall take place after the registration to IMS core.



The UE shall fetch the value of the Do-not-Disturb flag from the GLMS. This is done in the first GET request sent to GLMS (1-2). The UE may then download or update local version of access lists (3) (see 7.7.2.1), contact lists (4) (see 7.7.1.2) and groups (5) (see 7.7.3.2). The order of performing these operations may be different than shown in this scenario.

## 8 Im – Is correlation issues

List management on the Im interface is in general used by the end users with the purpose to affect the behavior of the PoC service on the Is interface. Because different protocols are used on these interfaces, there is a need to correlate certain logical entities like user identities and group identities in order to be able to unambiguously address these entities on both interfaces. This chapter specifies the cases where the correlation is necessary as well as cases where it is not needed or even undesired.

### 8.1 Server addresses

The SIP URI and IP addresses of the PoC need not be correlated in any way with the HTTP URL and IP addresses of the GLMS.

### 8.2 User identities

The UE shall correlate the public user identity used on the Is interface with the user identity on the Im interface. Since SIP URI are used on both interfaces, the correlation is implicit.

NOTE: TEL URL may be used on the Is interface to invite participants to a session (dial-out). The user of the GLMS shall use his SIP URI to interact with GLMS over the Im interface.

### 8.3 Contact list identities

Contact lists are addressed on both the Is and the Im interface by their SIP URI. On Im interface the identity of the contact list is included inside the request URI of the HTTP GET request, i.e. it is not used as the request URI itself.

### 8.4 Group identities

Groups are addressed on both the Is and the Im interface by their SIP URI. On Im interface the identity of the group is included inside the request URI of the HTTP GET request, i.e. it is not used as the request URI itself.

## 8.5 Authentication schemes

For the HTTP digest authentication scheme, the UE uses the Private User Identity and the password configured in the UE to authenticate its HTTP requests with the GLMS.

### Annex A (normative): UE configuration parameters for list management

Table 1 details the minimum set of parameters that shall be configurable in the UE in order for the UE to be able to communicate with the GLMS.

Parameter	Explanation	Type (Note 1 & 3)	Status	Examples (Note 2)
GLMSAddress	URI address of the GLMS. The URI scheme depends on the used transport binding.	string[250]	mandatory	“http://glms.operator.com:8080/glms” “https://130.100.78.43:8080”
SipPublicUserID	User’s SIP URI, parameter from SIP part of the client application shall be reused. See reference [3].	string[250]	mandatory	“sip:joe.doe@my-operator.com”
SipPrivateUserID	Parameter from the SIP part of the client shall be reused. See reference [3].	string[250]	mandatory	“u29502566@my-operator.com”
Proxy	Host name and optionally port of a HTTP proxy depending on the used transport binding.	String[120]	optional	“www.proxy.operator.com:1080”
<p>Note 1. The notation “string[]” designates a string of characters in UTF-8 encoding (see [11]) of the maximum length specified in the square brackets.</p> <p>Note 2. The example strings are shown between double quotation marks. The quotation marks are not part of the string.</p> <p>Note 3. The strings containing a URI shall not contain any characters reserved in the URI scheme. The reserved characters shall be represented in the escaped encoding as according to the URI scheme.</p>				

**Table 1 UE configuration parameters for list management**

---

## Annex B (informative): Change history

Date	Subject/Comment	Old	New
2004-05-12	Agreed by Comneon, Ericsson, Motorola, Nokia, Siemens		2.0.5
2004-06-07	Removed "confidential" and "proprietary" notes and updated change history	2.0.5	2.0.6