3GPP TSG-CN Meeting #26
Athens, Greece. 8th to 10th December 2004.

Tdoc ⌘NP-040593

CR-Form-v7.1

CHANGE REQUEST

⌘ **24.141** **CR** **21** ⌘ **rev** **1** ⌘ Current version: **6.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: | UICC apps⌘ ☐ ME ☒ Radio Access Network ☐ Core Network ☒

Title:	⌘	Introduction of XCAP client and XCAP server
Source:	⌘	Siemens
Work item code:	⌘	PRESNC
Date:	⌘	07/12/2004
Category:	⌘	**F**
Release:	⌘	Rel-6

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

Ph2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*
Rel-7	*(Release 7)*

Reason for change:	⌘	24.141 defines the roles Data Manipulator DM and Data Manipulation Server. However, in the XCAP drafts the terms ìXCAP clientî and ìXCAP serverî are defined and used for that purpose. As already discussed in the last meeting, the terms DM and DMS shall be replaced by XCAP client and XCAP server
Summary of change:	⌘	See reason for change
Consequences if not approved:	⌘	24.141 defines roles that are already defined in IETF, ambiguity

Clauses affected:	⌘	3.2, 4, 6, A.8			
Other specs affected:	⌘	**Y**	**N**		⌘
			X	Other core specifications	
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘				

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.htm.
Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be

downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

*** 1st change ***

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AS	Application Server
AUID	Application Usage ID
CN	Core Network
CPIM	Common Profile for Instant Messaging
CSCF	Call Session Control Function
~~DM~~	~~Data Manipulator~~
~~DMS~~	~~Data Manipulation Server~~
EPA	Event Publication Agent
ESC	Event State Compositor
HSS	Home Subscriber Server
HTTP	HyperText Transfer Protocol
I-CSCF	Interrogating - CSCF
IM	IP Multimedia
IOI	Inter Operator Identifier
IP	Internet Protocol
MIME	Multipurpose Internet Mail Extensions
P-CSCF	Proxy - CSCF
PIDF	Presence Information Data Format
PNA	Presence Network Agent
PS	Presence Server
PSI	Public Service Identity
PUA	Presence User Agent
RLMI	Resource List Meta-Information
RLS	Resource List Server
RPID	Rich Presence Information Data
S-CSCF	Serving - CSCF
SIP	Session Initiation Protocol
TLS	Transport Layer Security
UE	User Equipment
URI	Universal Resource Identifier
XCAP	XML Configuration Access Protocol
XML	Extensible Markup Language

**** next change ****

4 Presence service overview

The presence service provides the ability for the home network to manage presence information of a user's device, service or service media even whilst roaming. A user's presence information may be obtained through input from the user, information supplied by network entities or information supplied by elements external to the home network. Consumers of presence information, watchers, may be internal or external to the home network. The architecture for the 3GPP presence service is specified in 3GPP TS 23.141 [4].

SIP and XCAP provide means to manipulate the presence status of a user. For details on the differences between those means refer to draft-ietf-sip-publish-03 [23] and draft-isomaki-simple-xcap-pidf-manipulation-usage-00 [34]. For details on the relationship of ~~DMS~~ XCAP server to other roles see subclause 6.2.2.

Editor's note: It may be appropriate to include text in this clause pointing to the stage 1 on group management.

**** next change ****

6 Protocol for data manipulation at the Ut reference point

6.1 Introduction

Hypertext Transfer Protocol (HTTP) and XML Configuration Access Protocol (XCAP) are used to store, alter and delete data related to the presence service. The general information that can be manipulated is user groups, subscription authorization policy, resource lists, hard state presence publication, MIME objects referenced from the hard state presence information, etc. Soft state presence information manipulated with a PUBLISH request is not manipulated by the mechanism provided over the Ut reference point.

6.2 Functional entities

6.2.1 User Equipment (UE)

The UE implements the ~~Data Manipulator (DM)~~XCAP client role as described in subclause 6.3.1.

The UE shall implement HTTP digest AKA (see RFC 3310 [20]) and it shall initiate a bootstrapping procedure with the bootstrapping server function located in the home network, as described in 3GPP TS 24.109 [7].

The UE shall acquire the subscriber's certificate from PKI portal by using a bootstrapping procedure, as described in 3GPP TS 24.109 [7].

The UE shall implement HTTP digest authentication (see RFC 2617 [15A]).

The UE shall implement Transport Layer Security (TLS) (see RFC 2246 [13]). The UE shall be able to authenticate the network application function based on the received certificate during TLS handshaking phase.

6.2.2 Application Server (AS)

If an AS implements the role of a PS (see subclause 5.3.3) or of a RLS (see subclause 5.3.4), then the AS shall also implement the role of a ~~Data Manipulation Server (DMS)~~XCAP server (see subclause 6.3.2).

If there is no authentication proxy in the network, then the AS shall:

1) implement the role of a network application function, as described in 3GPP TS 24.109 [7];

2) implement TLS (see RFC 2246 [13]);

implement HTTP digest authentication (see RFC 2617 [15A]); and

4) support certificate authentication.

Editor's note: It needs to be clarified what physical entities can contain the Authentication Proxy and its relationship with the IMS architecture.

6.2.3 Authentication proxy

The authentication proxy shall implement the role of a network application function, as described in 3GPP TS 24.109 [7] and it shall support HTTP Digest Authentication (see RFC 2617 [15A] and certificate authentication.

The Authentication Proxy shall authenticate the UE and integrity protect the messages sent towards the UE.

Editor's note: It is FFS how the Authentication Proxy passes the user's identity to the Application Server (AS).

6.3 Roles

6.3.1 ~~Data Manipulator (DM)~~XCAP client

6.3.1.1 Introduction

The ~~DM~~ XCAP client is a logical function ~~that implements the requirements of a XCAP client~~ as defined in draft-ietf-simple-xcap-03 [33]. The ~~DM~~ XCAP client provides the means to manipulate the general data such us user groups, subscription authorization policy, resource lists, hard state presence publication, MIME objects referenced from the hard state presence information, etc.

NOTE: In order to be able to manipulate data stored on the ~~DMS~~XCAP server, the ~~DM~~ XCAP client has the root directory on the ~~DMS~~ XCAP server pre-configured or use some means to discover it. Discovery mechanisms are outside the scope of the present document.

6.3.1.2 Manipulating a presencelist

When the ~~DM~~ XCAP client intends to manipulate a presencelist, it shall generate an HTTP PUT, GET or DELETE request in accordance with RFC 2616 [15], draft-ietf-simple-xcap-03 [33] and draft-ietf-simple-xcap-list-usage-02 [36].

6.3.1.3 Manipulating the subscription authorization policy

When the ~~DM~~ XCAP client intends to manipulate the subscription authorization policy, it shall generate an HTTP PUT, HTTP GET or HTTP DELETE request in accordance with RFC 2616 [15], draft-ietf-simple-xcap-03 [33] and draft-ietf-simple-xcap-presence-rules-00 [35].

The ~~DM~~ XCAP client may use an HTTP GET in accordance with RFC 2616 [15], draft-ietf-simple-xcap-03 [33] and draft-rosenberg-simple-common-policy-caps-01 [42] for fetching of the authorization policy capabilities which the ~~DMS~~ XCAP server supports.

When the ~~DM~~ XCAP client intends to authorize a different value of the same presence attribute to different watchers or watcher groups, the ~~DM~~ XCAP client shall authorize a single tuple including one of the different values of the same presence attribute to every watcher or watcher groups by using a specific "inclusion set" as specified in draft-ietf-simple-xcap-presence-rules-00 [35].

6.3.1.4 Publishing hard state presence information

The ~~DM~~ XCAP client shall implement draft-isomaki-simple-xcap-pidf-manipulation-usage-00 [34] in order to be able to manipulate hard state presence information. Hard state presence information uses the same format as soft state information, namely "application/pidf+xml" content type as described in draft-ietf-impp-cpim-pidf-08 [21] together with any of its extensions.

When the hard state presence information contains one ore more MIME objects to be aggregated with the "application/pidf+xml" content type and any of its extensions, the ~~DM~~ XCAP client shall:

a) construct as many HTTP URIs as many objects to be stored and formulate every HTTP URI according a predefined directory structure;

NOTE: In order to be able to manipulate data stored on the ~~DMS~~XCAP server, the ~~DM~~ XCAP client has the root directory on the ~~DMS~~ XCAP server pre-configured or use some means to discover it. Discovery mechanisms are outside the scope of the present document.

b) store the objects on the data manipulation server behind the HTTP URI(s) created in the previous step using standard HTTP procedures as defined in RFC 2616 [15];

c) include every HTTP URI as a value of the corresponding XML element in the published "application/pidf+xml" presence document referencing the stored object(s) in the previous step; and

d) publish the hard state presence information according to draft-isomaki-simple-xcap-pidf-manipulation-usage-00 [34].

6.3.2 ~~Data Manipulation Server (DMS)~~XCAP server

6.3.2.1 Introduction

The ~~Data Manipulation Server (DMS)~~XCAP server is a logical function ~~that implements the requirements of a XCAP server~~ as defined in draft-ietf-simple-xcap-03 [33]. The ~~DMS~~ XCAP server can store data such us user groups, subscription authorization policy, resource lists, hard state presence information, MIME objects referenced from the hard state presence information, etc.

6.3.2.2 Resource list manipulation acceptance

When the data manipulation server receives an HTTP PUT, HTTP GET or HTTP DELETE request for manipulating or fetching a resource list, the ~~DMS~~ XCAP server shall first authenticate the request in accordance with 3GPP TS 24.109 [7] and then perform authorization. Afterwards the ~~DMS~~ XCAP server shall perform the requested action and generate a response in accordance with RFC 2616 [15], draft-ietf-simple-xcap-03 [33] and draft-ietf-simple-xcap-list-usage-02 [36].

6.3.2.3 Subscription authorization policy manipulation acceptance

When the ~~DMS~~ XCAP server receives an HTTP PUT, HTTP GET or HTTP DELETE request for manipulating or fetching of the subscription authorization policy, the data manipulation server shall first authenticate the request in accordance with 3GPP TS 24.109 [7] and then perform authorization. Afterwards the ~~DMS~~ XCAP server shall perform the requested action and generate a response in accordance with RFC 2616 [15], draft-ietf-simple-xcap-03 [33] and draft-ietf-simple-xcap-presence-rules-00 [35].

When the ~~DMS~~ XCAP server receives an HTTP GET request for fetching of the authorization policy capabilities information, the ~~DMS~~ XCAP server shall generate a response in accordance with RFC 2616 [15], draft-ietf-simple-xcap-02 [33] and draft-rosenberg-simple-pres-policy-caps-00 [42].

6.3.2.4 Publication acceptance of hard state presence information

When the ~~DMS~~ XCAP server receives an HTTP PUT, HTTP GET or HTTP DELETE request for publishing, fetching or deleting of hard state presence information, the ~~DMS~~ XCAP server shall first authenticate the request in accordance with 3GPP TS 24.109 [7] and then perform authorization. Afterwards the ~~DMS~~ XCAP server shall:

a) if the HTTP URI points to a predefined directory reserved for storing MIME objects and the request is an HTTP PUT request, replace any existing content referenced by the Request-URI with the content of the request;

b) if the Request-URI points to an uncreated directory, create the directory, store the content there and associate the content with the Request-URI. For all requests, i.e. HTTP PUT, HTTP GET and HTTP DELETE requests, generate an appropriate response in accordance with RFC 2616 [15]; or

c) if the HTTP URI points to an XCAP directory and the Application Usage ID (AUID) part of the HTTP URI is set to "pidf-manipulation", process the request and generate an appropriate response in accordance with draft-ietf-simple-xcap-03 [33], draft-isomaki-simple-xcap-pidf-manipulation-usage-00 [34] and RFC 2616 [15].

*** next change****

A.8 Example signalling flows of HTTP based presence service operation

A.8.1 Introduction

This subclause shows signalling flows relating to the manipulation of presence service data over the Ut reference point using XCAP.

Each example signalling flow shows several sequences of manipulation of data for the presence service.

NOTE: Error conditions are not considered in the examples e.g. if authorization checks fail in the XCAP server, XML Schema compliancy checks fail or the file specified by the URI does not exist then an appropriate 4xx response is sent to the client.

Editor's note: Clarifications how XCAP is using HTTP is needed.

A.8.2 Signalling flows demonstrating how ~~DMs~~ XCAP clients manipulate resource lists

Editor's note: The possible proxies (e.g., handling authentication matters) between the data manipulator client and HTTP server are bypassed. Also the authentication related headers are missing.

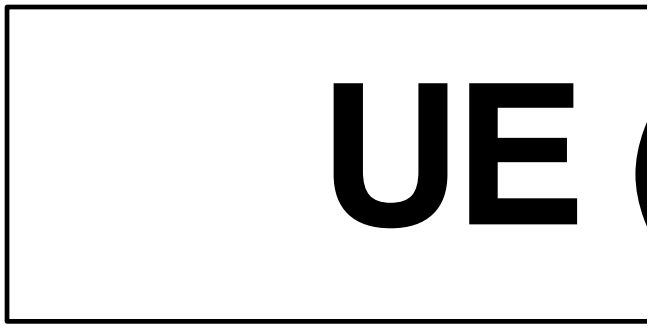

Figure A.8.2-1: ~~DM~~ XCAP client manipulating a resource list on ~~DMS~~XCAP server

Figure A.8.2-1 shows a how a ~~DM~~ XCAP client may manipulate a resource list on a ~~DMS~~XCAP server. The details of the signalling flows are as follows:

1. **XCAP PUT request (~~DM~~ XCAP client to ~~DMS~~ XCAP server - see example in table A.8.2-1**

The ~~DM~~ XCAP client generates an XCAP PUT request to create a new resource list on the ~~DMS~~XCAP server. The resource list has one entry.

Table A.8.2-1: XCAP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
PUT http://xcap.home1.net/services/resource-lists/users/user1/pf.xml HTTP/1.1
User-Agent: IMS subscriber
Referer: http://oper.home1.net:1234/service
Date: Thu, 08 Jan 2004 10:13:17 GMT
Content-Type: application/resource-lists+xml
Content-Length: (…)

<?xml version="1.0" encoding="UTF-8"?>
   <resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
     <list name="Presence_fellows" uri="sip:user1_list1@home1.net" subscribeable="true">
       <entry name="user2" uri="sip:user2_public1@home2.net">
         <display-name>User2</display-name>
       </entry>
     </list>
   </resource-lists>
```

2. **XCAP 201 (Created) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) ñ see example in table A.8.2-2**

After the ~~DMS~~ XCAP server has performed the necessary authorization checks on the originator to ensure the ~~DM~~ XCAP client is allowed to create a file, the ~~DMS~~ XCAP server sends an XCAP 201 (Created) response to the ~~DM~~XCAP client.

Table A.8.2-2: XCAP 201 (Created) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 201 CREATED
Server: Apache/1.3.22 (Unix) mod_perl/1.27
Date: Thu, 08 Jan 2004 10:50:35 GMT
Content-Type: text/html
Content-Length: 0
```

3. **XCAP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server) ñ see example in table A.8.2-3**

The ~~DM~~ XCAP client adds a new entry to the previously created resource list by generating a new XCAP PUT request.

Table A.8.2-3: XCAP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
PUT http://xcap.home1.net/services/resource-lists/users/user1/pf.xml?resource-
lists/list[@name="Presence_fellows"]/entry HTTP/1.1
User-Agent: IMS subscriber
Referer: http://oper.home1.net:1234/service
Date: Thu, 08 Jan 2004 10:13:17 GMT
Content-Type: application/xml-fragment-body
Content-Length: (…)

<?xml version="1.0" encoding="UTF-8"?>
         <entry name="user3" uri="sip:user3_public1@home3.net">
           <display-name>User3</display-name>
         </entry>
```

4. **XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) - see example in table A.8.2-4**

After the ~~DMS~~ XCAP server has performed the necessary authorization checks, XML document validations and XML schema compliancy checks the ~~DMS~~ XCAP server sends an XCAP 201 (Created) response to the ~~DM~~XCAP client.

Table A.8.2-4: XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 200 OK
Server: Apache/1.3.22 (Unix) mod_perl/1.27
```

```
Date: Thu, 08 Jan 2004 10:50:45 GMT
Content-Type: text/html
Content-Length: 0
```

5. **XCAP DELETE request (~~DM~~ XCAP client to ~~DMS~~XCAP server) - see example in table A.8.2-5**

The ~~DM~~ XCAP client decides to delete the entry "user2" from the resource list. The ~~DM~~ XCAP client generates an XCAP DELETE request.

Table A.8.2-5: XCAP DELETE request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
DELETE http://xcap.home1.net/services/resource-lists/users/user1/pf.xml?resource-
lists/list[@name="Presence_fellows"]/entry[@name=user2"] HTTP/1.1
Host: oper.example.com:9999
User-Agent: IMS subscriber
Date: Thu, 08 Jan 2004 10:14:17 GMT
Referer: http://oper.home1.net:1234/service
```

6. **XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) ñ see example in table A.8.2-6**

After the ~~DMS~~ XCAP server has performed the necessary authorization checks on the originator to ensure the ~~DM~~ XCAP client is allowed to delete an entry from the resource list, the ~~DMS~~ XCAP server sends an XCAP 200 (OK) response.

Table A.8.2-6: XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 200 OK
Server: Apache/1.3.22 (Unix) mod_perl/1.27
Date: Thu, 08 Jan 2004 11:00:47 GMT
Content-Type: image/jpeg
Content-Length: 0
```

7. **XCAP GET request (~~DM~~ XCAP client to ~~DMS~~XCAP server) ñ see example in table A.8.2-7**

The ~~DM~~ XCAP client wishes to check the result of the previous transaction by generating an XCAP GET request.

Table A.8.2-7: XCAP GET request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
GET http://xcap.home1.net/services/resource-lists/users/user1/pf.xml HTTP/1.1
User-Agent: IMS subscriber
Referer: http://oper.home1.net:1234/service
Date: Thu, 08 Jan 2004 11:13:17 GMT
Content-Length: 0
```

8. **XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) - see example in table A.8.2-8**

After the ~~DMS~~ XCAP server has performed the necessary authorization checks on the originator to ensure the ~~DM~~ XCAP client is allowed to fetch the resource list, the ~~DMS~~ XCAP server sends an XCAP 200 (OK) response to the ~~DM~~ XCAP client including the resource list in the body of the response.

Table A.8.2-8: XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 200 OK
Server: Apache/1.3.22 (Unix) mod_perl/1.27
Etag: "askdajdsaj"
Date: Thu, 08 Jan 2004 11:50:35 GMT
Content-Type:application/resource-lists+xml
Content-Length: (…)

<?xml version="1.0" encoding="UTF-8"?>
   <resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
     <list name="Presence_fellows" uri="sip:user1_list1@home1.net" subscribeable="true">
       <entry name="user3" uri="sip:user3_public1@home3.net">
         <display-name>User3</display-name>
       </entry>
```

```
    </list>
  </resource-lists>
```

A.8.3 Signalling flows demonstrating how ~~DMs~~ XCAP clients manipulate presence authorization policy

Editor's note: The possible proxies (e.g., handling authentication matters) between the data manipulator client and HTTP server are bypassed. Also the authentication related headers are missing.

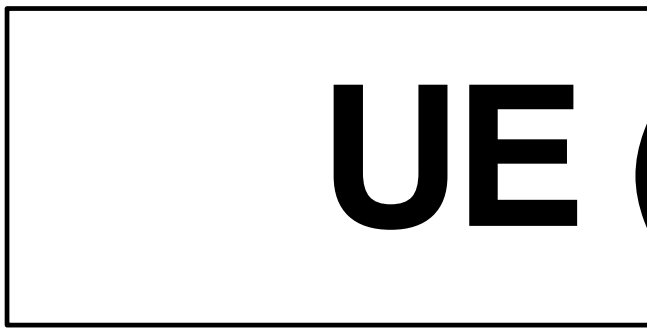

Figure A.8.3-1: ~~DM~~ XCAP client manipulating presence authorization policy on ~~DMS~~XCAP server

Figure A.8.3-1 shows a ~~DM~~ XCAP client manipulating presence authorization policy on a ~~DMS~~XCAP server. The details of the signalling flows are as follows:

1. **XCAP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server) ñ see example in table A.8.3-1**

   The ~~DM~~ XCAP client generates an XCAP PUT request to create a presence authorization policy on the ~~DMS~~XCAP server. The presence authorization policy has one permission statement allowing for sip:user2_public1@home2.net to see all information from the basic PIDF namespace along with the "video" element from the prescaps namespace.

Table A.8.3-1: XCAP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
PUT http://xcap.home1.net/services/permission-statements/users/user1/ps.xml HTTP/1.1
User-Agent: IMS subscriber
Referer: http://oper.home1.net:1234/service
Date: Thu, 08 Jan 2004 10:13:17 GMT
Content-Type:application/permission-statements+xml
Content-Length: (…)

<?xml version="1.0" encoding="UTF-8"?>
   <permission-statements xmlns="urn:ietf:params:xml:ns:permission-statements"
     xmlns:pidf="urn:ietf:params:xml:ns:pidf"
     xmlns:prescaps="urn:ietf:params:xml:ns:simple-prescaps-ext">
```

```
    <statement id="dsafa43232">
      <applies-to>
        <uri>sip:user2_public1@home2.net</uri>
      </applies-to>

      <permissions>
        <accept/>
        <show-namespace>urn:ietf:params:xml:ns:pidf</show-namespace>
        <show-element>prescaps:video</show-element>
      </permissions>

    </statement>
  </permission-statements>
```

2. **XCAP 201 (Created) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) - see example in table A.8.3-2**

After the ~~DMS~~ XCAP server has performed the necessary authorization checks on the originator to ensure the ~~DM~~ XCAP client is allowed to create a file, the ~~DMS~~ XCAP server sends an XCAP 201 (Created) response to the ~~DM~~XCAP client.

Table A.8.3-2: XCAP 201 (Created) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 201 CREATED
Server: Apache/1.3.22 (Unix) mod_perl/1.27
Date: Thu, 08 Jan 2004 10:50:35 GMT
Content-Type: text/html
Content-Length: 0
```

3. **XCAP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server) ñ see example in table A.8.3-3**

The ~~DM~~ XCAP client adds a new permission-statement to the previously created presence authorization policy by generating a new XCAP request. The new permission statement allows the user named sip:user3_public1@home3.net to see the tuple with class element specifying "sip".

Table A.8.3-3: XCAP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
PUT http://xcap.home1.net/services/permission-statements/users/user1/ps.xml/permission-
      statements HTTP/1.1
User-Agent: IMS subscriber
Referer: http://oper.home1.net:1234/service
Date: Thu, 08 Jan 2004 10:13:27 GMT
Content-Type: application/xml-fragment-body
Content-Length: (…)

<?xml version="1.0" encoding="UTF-8"?>
    <statement id="dsffdsfrrr32423">
      <applies-to>
        <uri>sip:user3_public1@home3.net</uri>
      </applies-to>

      <permissions>
        <accept/>
        <show-tuple>sip</show-tuple>
      </permissions>

    </statement>
```

4. **XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) - see example in table A.8.3-4**

After the ~~DMS~~ XCAP server has performed the necessary authorization checks, XML document validations and XML schema compliancy checks the ~~DMS~~ XCAP server sends an XCAP 201 (Created) response to the ~~DM~~XCAP client.

Table A.8.3-4: XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 200 OK
Server: Apache/1.3.22 (Unix) mod_perl/1.27
Date: Thu, 08 Jan 2004 10:50:45 GMT
```

```
Content-Type: text/html
Content-Length: 0
```

5. **XCAP DELETE request (~~DM~~ XCAP client to ~~DMS~~XCAP server) - see example in table A.8.3-5**

The ~~DM~~ XCAP client decides to delete the permission-statement for sip:user2_public1@home2.net from the authorization policy. The ~~DM~~ XCAP client generates an XCAP DELETE request.

Table A.8.3-5: XCAP DELETE request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
DELETE http://xcap.home1.net/services/presence-lists/users/user1/ps.xml/permission-
      statements/statement[@id="dsafa43232"]/permissions/show-namespace HTTP/1.1
Host: oper.example.com:9999
User-Agent: IMS subscriber
Date: Thu, 08 Jan 2004 10:14:17 GMT
Referer: http://oper.home1.net:1234/service
```

6. **XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) ñ see example in table A.8.3-6**

After the ~~DMS~~ XCAP server has performed the necessary authorization checks on the originator to ensure the ~~DM~~ XCAP client is allowed to delete an entry from the resource list, the ~~DMS~~ XCAP server sends an XCAP 200 (OK) response.

Table A.8.3-6: XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 200 OK
Server: Apache/1.3.22 (Unix) mod_perl/1.27
Date: Thu, 08 Jan 2004 11:00:47 GMT
Content-Length: 0
```

7. **XCAP GET request (~~DM~~ XCAP client to ~~DMS~~XCAP server) ñ see example in table A.8.3-7**

The ~~DM~~ XCAP client wishes to check the result of the previous transaction by generating an XCAP GET request.

Table A.8.3-7: XCAP GET request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
GET http://xcap.home1.net/services/permission-statements/users/user1/ps.xml HTTP/1.1
User-Agent: IMS subscriber
Referer: http://oper.home1.net:1234/service
Date: Thu, 08 Jan 2004 11:13:17 GMT
Content-Length: 0
```

8. **XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) ñ see example in table A.8.3-8**

After the ~~DMS~~ XCAP server has performed the necessary authorization checks on the originator to ensure the ~~DM~~ XCAP client is allowed to fetch the resource list, the ~~DMS~~ XCAP server sends an XCAP 200 (OK) response to the ~~DM~~ XCAP client including the resource list in the body of the response.

Table A.8.3-8: XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 200 OK
Server: Apache/1.3.22 (Unix) mod_perl/1.27
Etag: "eiuuekksks"
Date: Thu, 08 Jan 2004 11:50:35 GMT
Content-Type:application/permission-statements+xml
Content-Length: (…)

<?xml version="1.0" encoding="UTF-8"?>
   <permission-statements xmlns="urn:ietf:params:xml:ns:permission-statements"
     xmlns:pidf="urn:ietf:params:xml:ns:pidf"
     xmlns:prescaps=" urn:ietf:params:xml:ns:simple-prescaps-ext">
     <statement id="dsffdsfrrr32423">
       <applies-to>
         <uri>sip:user3_public1@home3.net</uri>
       </applies-to>
       <permissions>
         <accept/>
         <show-tuple>sip</show-tuple>
       </permissions>
     </statement>
   </permission-statements>
```

A.8.4 Storing external content (successful operation)

Editor's note: The possible proxies (e.g., handling authentication matters) between the data manipulator client and HTTP server are bypassed. Also the authentication related headers are missing.

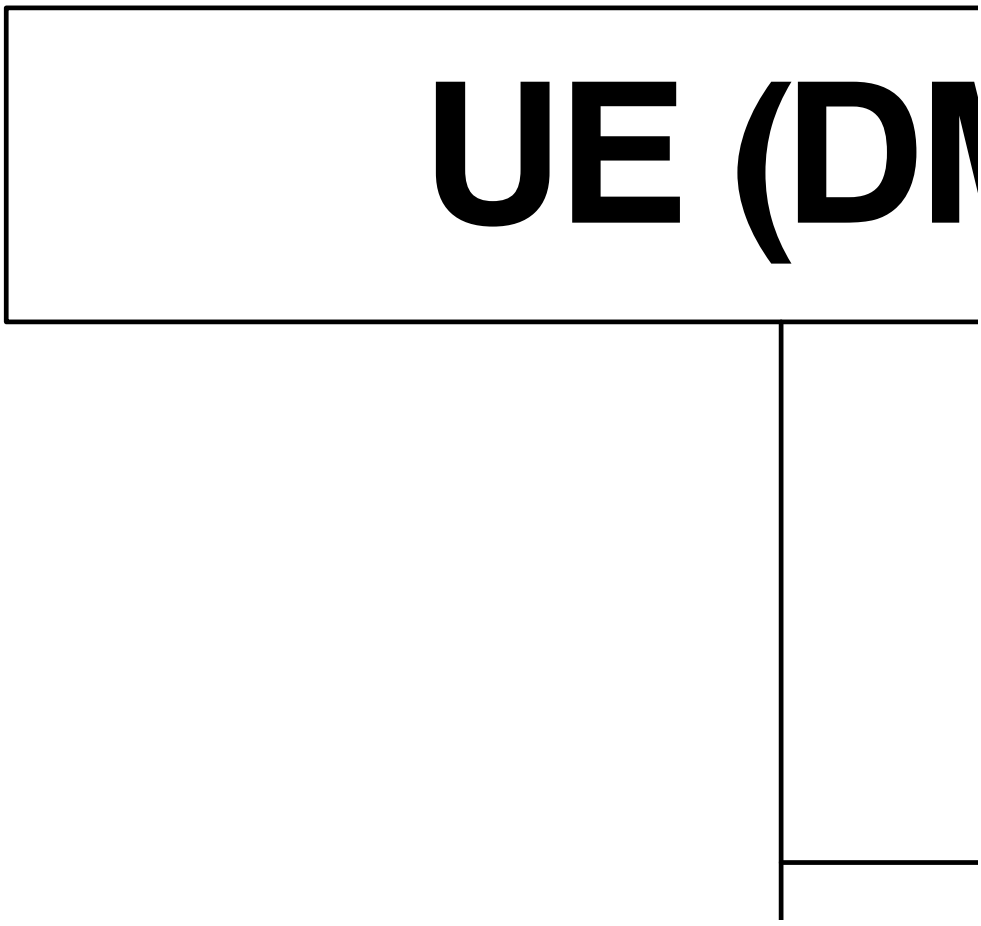

Figure A.8.4.-1: ~~DM~~ XCAP client manipulating hard-state presence document on ~~DMS~~XCAP server

Figure A.8.4-1 shows a ~~DM~~ XCAP client manipulating hard-state presence document on a ~~DMS~~ XCAP server when the presence document has an aggregated storing MIME object with the "application/pidf+xml" content type and any of its extensions. The details of the signalling flows are as follows:

1. **HTTP PUT request (~~DM~~ XCAP client ~~(client)~~ to ~~DMS~~XCAP server) ñ see example in table A.8.2-1**

   In order to store the content, the ~~DM~~ XCAP client generates an HTTP PUT request containing the MIME object in the body of the request. The request-URI points to the directory where the content is stored and shows the name of the file to be created.

Table A.8.4-1: HTTP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
PUT http://operator.example.com/services/users/bill/pictureX HTTP/1.1
User-Agent: IMS subscriber
Referer: http://oper.home1.net:1234/service
Date: Thu, 08 Jan 2004 10:13:17 GMT
Content-Type: image/jpeg
Content-Length: (…)

{pictureX.jpg}
```

2. **HTTP 201 (Created) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) ñ see example in table A.8.4-2**

   After the ~~DMS~~ XCAP server has performed the necessary authorization checks on the originator to ensure the ~~DM~~ XCAP client is allowed to create a file the HTTP server sends an HTTP 201 (Created) response to the client.

Table A.8.4-2: HTTP 201 (Created) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 201 CREATED
Server: Apache/1.3.22 (Unix) mod_perl/1.27Content-Type: text/html
Date: Thu, 08 Jan 2004 10:50:35 GMT
Content-Length: 1234
```

3. **XCAP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server) - see example in table A.8.2-3**

   The ~~DM~~ XCAP client generates an XCAP PUT request in order to store XML encoded presence document which includes a URI reference to the MIME object stored on the ~~DMS~~XCAP server. The AUID part of the request URI is 'pidf-manipulation' as defined in draft-isomaki-simple-xcap-pidf-manipulation-usage-00 [34].

Table A.8.4-3: XCAP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
PUT http://xcap.example.com/services/pidf-manipulation/users/bill/pidf.xml HTTP/1.1
User-Agent: IMS subscriber
Referer: http://xcap.home1.net:1234/service
Date: Thu, 08 Jan 2004 10:13:27 GMT
Content-Type: application/pidf+xml
Content-Length: (…)

<?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:cpim-pidf"
          xmlns:et="urn:ietf:params:xml:ns:pidf:rpid-tuple"
          xmlns:ext="urn:ietf:params:xml:ns:ext-cont"
          entity="sip:bill@example.com">

     <tuple id="123sd">
       <status>
         <basic>open</basic>
       </status>
       <et:type>service</et:type>
         <contact>sip:bill@example.com</contact>
     </tuple>

     <tuple id="432sd">
       <status>
         <basic>open</basic>
       </status>
       <et:type>presentity</et:type>
       <ext:photo>
```

```
          http://operator.example.com/services/users/bill/pictureX.jpg
        </ext:photo>
        <note xml:lang="en">At home</note>
      </tuple>
    </presence>
```

4. **XCAP 201 (CREATED) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) - see example in table A.8.4-4**

After the ~~DMS~~ XCAP server has performed the necessary authorization checks, XML document validations and XML schema compliancy checks the ~~DMS~~ XCAP server sends an XCAP 201 (Created) response to the ~~DM~~XCAP client.

Table A.8.4-4: XCAP 201 (Created) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 201 CREATED
Server: Apache/1.3.22 (Unix) mod_perl/1.27
Date: Thu, 08 Jan 2004 10:50:45 GMT
Content-Length: 0
```

5. **HTTP GET request (~~DM~~ XCAP client to ~~DMS~~XCAP server) ñ see example in table A.8.4-5**

The ~~DM~~ XCAP client wishes to fetch the MIME object from the ~~DMS~~XCAP server. The client generates an HTTP GET request. The request URI points to the directory where the object is stored and indicates the name of the file to be fetched.

Table A.8.4-5: HTTP GET request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
GET http://operator.example.com/services/users/bill/pictureX HTTP/1.1
Host: oper.example.com:9999
User-Agent: IMS subscriber
Date: Thu, 08 Jan 2004 10:43:17 GMT
Accept: image/jpeg
Referer: http://oper.home1.net:1234/service
```

6. **HTTP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) ñ see example in table A.8.4-6**

After the ~~DMS~~ XCAP server has performed the necessary authorization checks on the originator to ensure the ~~DM~~ XCAP client is allowed to fetch the file the ~~DMS~~ XCAP server sends an HTTP 200 (OK) response having the object in the body to the ~~DM~~XCAP client.

Table A.8.4-6: HTTP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 200 OK
Server: Apache/1.3.22 (Unix) mod_perl/1.27
Date: Thu, 08 Jan 2004 11:00:47 GMT
Content-Type: image/jpeg
Content-Length: (…)

{pictureX}
```

7. **HTTP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server) ñ see example in table A.8.4-7**

The ~~DM~~ XCAP client wishes to modify the earlier stored MIME object by replacing the picture X with a new picture X with new content. To modify the object the ~~DM~~ XCAP client generates an HTTP PUT request using the same request URI as has been used for the modified (old) object. The new object is conveyed in the body of the request.

Table A.8.4-7: HTTP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
PUT http://operator.example.com/services/users/bill/pictureX HTTP/1.1
User-Agent: IMS subscriber
Referer: http://oper.home1.net:1234/service
Date: Thu, 08 Jan 2004 11:13:17 GMT
Content-Type: image/jpeg
Content-Length: (…)

{pictureX.jpg}
```

8. **HTTP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) ñ see example in table A.8.4-8**

After the ~~DMS~~ XCAP server has performed the necessary authorization checks on the originator to ensure the ~~DM~~ XCAP client is allowed to replace the existing MIME object with the new one the ~~DMS~~ XCAP server sends an HTTP 200 (OK) response to the ~~DM~~XCAP client.

Table A.8.4-8: HTTP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 200 OK
Server: Apache/1.3.22 (Unix) mod_perl/1.27
Date: Thu, 08 Jan 2004 11:50:35 GMT
Content-Length: 0
```

9. **XCAP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server) ñ see example in table A.8.4-9**

The ~~DM~~ XCAP client wishes to remove the MIME object from his presence information. The ~~DM~~ XCAP client generates an XCAP PUT request to modify the XML encoded presence document to remove the reference to the MIME object from the presence document. The request URI contains a node selector to the requested tuple according to draft-ietf-simple-xcap-02 [33]. Because the signalling flow does not contain the XCAP GET request the use of the If-Match header is omitted in this example.

Table A.8.4-9: XCAP PUT request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
PUT http://xcap.example.com/services/pidf-
  manipulation/users/bill/pidf.xml?presence/tuple[@id='432sd'] HTTP/1.1
Date: Thu, 08 Jan 2004 11:13:37 GMT
Content-Type: text/plain
Content-Length: (…)

<?xml version="1.0" encoding="UTF-8"?>
   <tuple id="432sd">
     <status>
       <basic>open</basic>
     </status>
     <et:type>presentity</et:type>
     <note xml:lang="en">At home</note>
   </tuple>
```

10. **XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) - see example in table A.8.4-10**

After the ~~DMS~~ XCAP server has performed the necessary authorization checks, XML document validations and XML Schema compliancy checks the ~~DMS~~ XCAP server sends an XCAP 200 (OK) response to the ~~DM~~XCAP client.

Table A.8.4-10: XCAP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 200 OK
Server: Apache/1.3.22 (Unix) mod_perl/1.27
Date: Thu, 08 Jan 2004 11:50:59 GMT
Content-Length: 0
```

11. **HTTP DELETE request (~~DM~~ XCAP client to ~~DMS~~XCAP server) ñ see example in table A.8.4-11**

The ~~DM~~ XCAP client removes the MIME object from the ~~DMS~~ XCAP server by generating an HTTP DELETE request.

Table A.8.4-11: HTTP DELETE request (~~DM~~ XCAP client to ~~DMS~~XCAP server)

```
DELETE http://operator.example.com/services/users/bill/pictureX HTTP/1.1
Host: oper.example.com:9999
User-Agent: IMS subscriber
Date: Thu, 08 Jan 2004 11:52:00 GMT
Referer: http://oper.home1.net:1234/service
```

12. **HTTP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client) ñ see example in table A.8.4-12**

After the ~~DMS~~ XCAP server has performed the necessary authorization checks on the originator to ensure that the ~~DM~~ XCAP client is allowed to delete the object, the ~~DMS~~ XCAP server sends an HTTP 200 (OK) response to the ~~DM~~XCAP client.

Table A.8.4-12: HTTP 200 (OK) response (~~DMS~~ XCAP server to ~~DM~~XCAP client)

```
HTTP/1.1 200 OK
Server: Apache/1.3.22 (Unix) mod_perl/1.27
Date: Thu, 08 Jan 2004 11:52:35 GMT
Content-Length: 0
```