

**Open Service Access (OSA);
Parlay X Web Services;
Part 14: Presence
(Parlay X 2)**



Reference

RES/TISPAN-01033-14-OSA

Keywords

API, OSA, service

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:

<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at

<http://portal.etsi.org/tb/status/status.asp>

If you find errors in the present document, please send your comment to one of the following services:

http://portal.etsi.org/chaicor/ETSI_support.asp

Copyright Notification

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2006.

© The Parlay Group 2006.

All rights reserved.

DECT™, **PLUGTESTS™** and **UMTS™** are Trade Marks of ETSI registered for the benefit of its Members.
TIPHON™ and the **TIPHON logo** are Trade Marks currently being registered by ETSI for the benefit of its Members.
3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.

Contents

| | |
|--|----|
| Intellectual Property Rights | 5 |
| Foreword..... | 5 |
| 1 Scope | 6 |
| 2 References | 6 |
| 3 Definitions and abbreviations..... | 7 |
| 3.1 Definitions | 7 |
| 3.2 Abbreviations | 7 |
| 4 Detailed service description | 8 |
| 4.1 Relationship to Similar or Supplanted Specifications | 8 |
| 5 Namespaces..... | 9 |
| 6 Sequence diagrams | 9 |
| 6.1 Interface flow overview..... | 9 |
| 7 XML Schema data type definition | 11 |
| 7.1 PresenceAttributeType enumeration | 11 |
| 7.2 ActivityValue enumeration | 11 |
| 7.3 PlaceValue enumeration..... | 12 |
| 7.4 PrivacyValue enumeration | 12 |
| 7.5 SphereValue enumeration | 12 |
| 7.6 CommunicationMeansType enumeration..... | 13 |
| 7.7 CommunicationMeans structure..... | 13 |
| 7.8 CommunicationValue structure..... | 13 |
| 7.9 OtherValue structure | 13 |
| 7.10 PresenceAttribute structure | 13 |
| 7.11 SubscriptionRequest structure..... | 14 |
| 7.12 PresencePermission structure | 14 |
| 8 Web Service interface definition..... | 14 |
| 8.1 Interface: PresenceConsumer | 14 |
| 8.1.1 Operation: subscribePresence | 14 |
| 8.1.1.1 Input message: subscribePresenceRequest..... | 15 |
| 8.1.1.2 Output message: subscribePresenceResponse..... | 15 |
| 8.1.1.3 Referenced faults..... | 15 |
| 8.1.2 Operation: getUserPresence..... | 15 |
| 8.1.2.1 Input message: getUserPresenceRequest | 15 |
| 8.1.2.2 Output message: getUserPresenceResponse | 15 |
| 8.1.2.3 Referenced faults..... | 16 |
| 8.1.3 Operation: startPresenceNotification | 16 |
| 8.1.3.1 Input message: startPresenceNotificationRequest..... | 16 |
| 8.1.3.2 Output message: startPresenceNotificationResponse..... | 16 |
| 8.1.3.3 Referenced faults..... | 17 |
| 8.1.4 Operation: endPresenceNotification | 17 |
| 8.1.4.1 Input message: endPresenceNotificationRequest..... | 17 |
| 8.1.4.2 Output message: endPresenceNotificationResponse..... | 17 |
| 8.1.4.3 Referenced faults..... | 17 |
| 8.2 Interface: PresenceNotification | 18 |
| 8.2.1 Operation: statusChanged..... | 18 |
| 8.2.1.1 Input message: statusChangedRequest..... | 18 |
| 8.2.1.2 Output message: statusChangedResponse..... | 18 |
| 8.2.1.3 Referenced faults..... | 18 |
| 8.2.2 Operation: statusEnd..... | 18 |
| 8.2.2.1 Input message: statusEndRequest | 18 |
| 8.2.2.2 Output message: statusEndResponse | 18 |

| | | |
|-------------------------------|---|-----------|
| 8.2.2.3 | Referenced faults..... | 18 |
| 8.2.3 | Operation: notifySubscription..... | 19 |
| 8.2.3.1 | Input message: notifySubscriptionRequest | 19 |
| 8.2.3.2 | Output message: notifySubscriptionResponse | 19 |
| 8.2.3.3 | Referenced faults..... | 19 |
| 8.2.4 | Operation: subscriptionEnded..... | 19 |
| 8.2.4.1 | Input message: subscriptionEndedRequest | 19 |
| 8.2.4.2 | Output message: subscriptionEndedResponse | 19 |
| 8.2.4.3 | Referenced faults..... | 19 |
| 8.3 | Interface: PresenceSupplier | 19 |
| 8.3.1 | Operation: publish | 19 |
| 8.3.1.1 | Input message: publishRequest | 20 |
| 8.3.1.2 | Output message: publishResponse | 20 |
| 8.3.1.3 | Referenced faults..... | 20 |
| 8.3.2 | Operation: getOpenSubscriptions | 20 |
| 8.3.2.1 | Input message: getOpenSubscriptionsRequest..... | 20 |
| 8.3.2.2 | Output message: getOpenSubscriptionsResponse..... | 20 |
| 8.3.2.3 | Referenced faults..... | 20 |
| 8.3.3 | Operation: updateSubscriptionAuthorization | 20 |
| 8.3.3.1 | Input message: updateSubscriptionAuthorizationRequest | 21 |
| 8.3.3.2 | Output message: updateSubscriptionAuthorizationResponse | 21 |
| 8.3.3.3 | Referenced faults..... | 21 |
| 8.3.4 | Operation: getMyWatchers..... | 21 |
| 8.3.4.1 | Input message: getMyWatchersRequest | 21 |
| 8.3.4.2 | Output message: getMyWatchersResponse | 21 |
| 8.3.4.3 | Referenced faults..... | 21 |
| 8.3.5 | Operation: getSubscribedAttributes..... | 22 |
| 8.3.5.1 | Input message: getSubscribedAttributesRequest | 22 |
| 8.3.5.2 | Output message: getSubscribedAttributesResponse | 22 |
| 8.3.5.3 | Referenced faults..... | 22 |
| 8.3.6 | Operation: blockSubscription | 22 |
| 8.3.6.1 | Input message: blockSubscriptionRequest..... | 22 |
| 8.3.6.2 | Output message: blockSubscriptionResponse..... | 22 |
| 8.3.6.3 | Referenced faults..... | 22 |
| 9 | Fault definitions..... | 23 |
| 9.1 | ServiceException..... | 23 |
| 9.1.1 | SVC0220: No subscription request..... | 23 |
| 9.1.2 | SVC0221: Not a watcher | 23 |
| 10 | Service policies | 23 |
| Annex A (normative): | WSDL for Presence | 24 |
| Annex B (informative): | Bibliography..... | 25 |
| History | | 26 |

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (<http://webapp.etsi.org/IPR/home.asp>).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN).

The present document is part 14 of a multi-part deliverable covering Open Service Access (OSA); Parlay X 2 Web Services, as identified below:

- Part 1: "Common";
- Part 2: "Third Party Call";
- Part 3: "Call Notification";
- Part 4: "Short Messaging";
- Part 5: "Multimedia Messaging";
- Part 6: "Payment";
- Part 7: "Account Management";
- Part 8: "Terminal Status";
- Part 9: "Terminal Location";
- Part 10: "Call Handling";
- Part 11: "Audio Call";
- Part 12: "Multimedia Conference";
- Part 13: "Address List Management";
- Part 14: "Presence".**

The present document has been defined jointly between ETSI, The Parlay Group (<http://www.parlay.org>) and the 3GPP.

The present document forms part of the Parlay X 2.1 set of specifications.

The present document is equivalent to 3GPP TS 29.199-14 V6.2.0 (Release 6).

1 Scope

The present document is part 14 of the Stage 3 Parlay X 2 Web Services specification for Open Service Access (OSA).

The OSA specifications define an architecture that enables application developers to make use of network functionality through an open standardized interface, i.e. the OSA APIs.

The present document specifies the Presence Web Service. The following are defined here:

- Name spaces.
- Sequence diagrams.
- Data definitions.
- Interface specification plus detailed method descriptions.
- Fault definitions.
- Service Policies.
- WSDL Description of the interfaces.

2 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 and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents which are not found to be publicly available in the expected location might be found at <http://docbox.etsi.org/Reference>.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

- [1] ETSI TR 121 905: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Vocabulary for 3GPP Specifications (3GPP TR 21.905 version 7.2.0 Release 7)".
- [2] W3C Recommendation (2 May 2001): "XML Schema Part 2: Datatypes".

NOTE: Available at <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>.

- [3] ETSI ES 202 391-1: "Open Service Access (OSA); Parlay X Web Services; Part 1: Common (Parlay X 2)".
- [4] ETSI ES 202 915-14: "Open Service Access (OSA); Application Programming Interface (API); Part 14: Presence and Availability Management SCF (Parlay 4)".
- [5] IETF RFC 3856: "A Presence Event Package for the Session Initiation Protocol (SIP)".
<http://www.ietf.org/rfc/rfc3856.txt>.
- [6] Void.
- [7] ETSI ES 202 391-13: "Open Service Access (OSA); Parlay X Web Services; Part 13: Address List Management (Parlay X 2)".

- [8] IETF RFC 3265: "Session Initiation Protocol (SIP)-Specific Event Notification".
- [9] Void.
- [10] ETSI ES 202 391-8: "Open Service Access (OSA); Parlay X Web Services; Part 8: Terminal Status (Parlay X 2)".
- [11] ETSI ES 202 391-9: "Open Service Access (OSA); Parlay X 2 Web Services; Part 9: Terminal Location (Parlay X 2)".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ES 202 391-1 [3] and the following apply:

applications: for Instant Messaging, Push to Talk, or call control and other purposes may become clients of the presence Web Service. We assume that these applications belong to a watcher and authenticate to the services in the name of the watcher

identity: represents a user in the real world

NOTE: See clause 4.4.1 in ES 202 915-14 [4].

presence attributes: contain information about a presentity. An attribute has a name and a value and can be supplied by any device, application or network module that can be associated to the presentity's identity. A watcher can obtain attributes only after he has successfully subscribed to them. Examples for attributes are activity, location type, communication means, etc.

presence information: set of attributes that characterize the presentity such as current activity, environment, communication means and contact addresses. Only the system and the presentity have direct access to this information, which may be collected and aggregated from **several** devices associated to the presentity

subscription: before a watcher can access presence data, he has to subscribe to it. One possibility the API provides is an end-to-end subscription concept, in which only identities that have accepted a subscription to their presence can be addressed. Subscriptions can be also automatically handled by server policies edited by the presentity or other authorized users. The service/protocol to manage those policies is out of the scope of the present document.

NOTE: This definition is not related to the term "subscription" in TR 121 905 [1].

watcher and presentity: names used to denote the role of the client connected to the presence services. As in Parlay/OSA PAM [4] the watcher and the presentity have to be associated to identities registered to the system, i.e. users, groups of users or organizations

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ES 202 391-1 [3] and the following apply:

| | |
|--------|--|
| IETF | Internet Engineering Task Force |
| IMS | IP Multimedia Subsystem |
| ISC | IP multimedia subsystem Service Control interface |
| MMS | Multimedia Message Service |
| PAM | Presence and Availability Management |
| SCF | Service Capability Feature |
| SIMPLE | SIP for Instant Messaging and Presence Leveraging Extensions |
| SIP | Session Initiation Protocol |
| SMS | Short Message Service |
| XCAP | XML Configuration Access Protocol |
| XML | eXtensible Markup Language |
| XMPP | eXtensible Messaging and Presence Protocol |
| XSD | XML Schema Definition |

4 Detailed service description

The presence service allows for presence information to be obtained about one or more users and to register presence for the same. It is assumed that the typical client of these interfaces is either a supplier or a consumer of the presence information. An Instant Messaging application is a canonical example of such a client of this interface.

Figure 1 shows the architecture of the Presence Web Service and the underlying services. The Parlay/OSA PAM SCF is the straightforward option and implements the presence server with extended identity, device capability, and presence agent management. Parlay/OSA PAM allows aggregation of presence information from internet, mobile and enterprise users, etc. using a presence transport network of SIP or XMPP servers. The Presence Web Service can however communicate directly for example with IMS presence network elements (presence and resource list servers) using the ISC (SIP/SIMPLE) protocol interface.

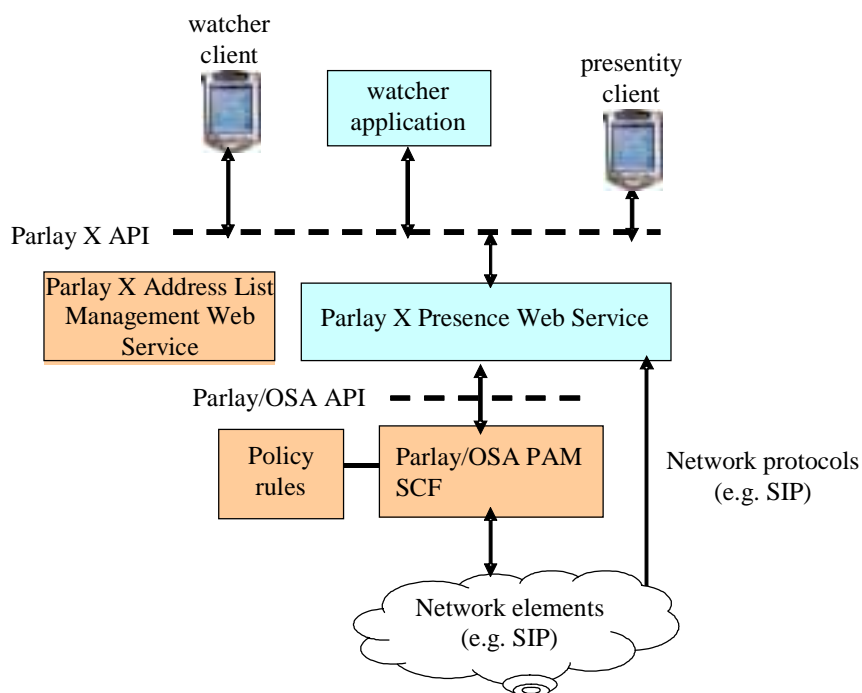


Figure 1: Presence web service environment

4.1 Relationship to Similar or Supplanted Specifications

The most important relations are to:

- Parlay X 2 Terminal Status Web Service [10] and Parlay X 2 Terminal Location Web Service [11]: Both services deal with information that could be considered part of the user's presence information. Communication abilities can be derived from terminal status information, and the user's placetype can be derived from his location.
- Parlay/OSA PAM [4]: The Parlay/OSA Presence and Availability specification can be considered the big brother of the present document. While Parlay X 2 Presence stays behind Parlay/OSA PAM in terms of flexibility and power - especially concerning attributes and management interfaces - it also extends PAM by introducing end-to-end authorization. The present document aims to be mappable to Parlay/OSA PAM.
- SIP SIMPLE [5]: The present document aims to be mappable to the SIP/SIMPLE architecture.
- XMPP (Jabber) (see Bibliography): Many principles of XMPP have been adopted, especially the end-to-end authorization.
- IETF Rich Presence (see Bibliography). The set of attributes the present document specifies is closely aligned with the IETF's Rich Presence ideas.

- Group Management [7]: Presence of groups is supported by the present document, however their creation and manipulation has to be done using the Parlay X 2 Address List Management Web Service. In the 3GPP presence context, contact lists and group manipulation is done with the XCAP protocol(see Bibliography).

5 Namespaces

The PresenceConsumer interface uses the namespace:

http://www.csapi.org/wsdl/parlayx/presence/consumer/v2_3

The PresenceNotification interface uses the namespace:

http://www.csapi.org/wsdl/parlayx/presence/notification/v2_3

The PresenceSupplier interface uses the namespace:

http://www.csapi.org/wsdl/parlayx/presence/supplier/v2_3

The data types are defined in the namespace:

http://www.csapi.org/schema/parlayx/presence/v2_3

The "xsd" namespace is used in the present document to refer to the XML Schema data types defined in XML Schema [2]. The use of the name "xsd" is not semantically significant.

6 Sequence diagrams

6.1 Interface flow overview

The sequence diagram shows the interactions in case both watcher application and presentity are Web Service clients. Compared to the SIP interactions, the subscription notification is separated from the delivery of presence information itself. Based on the subscription result, the watcher can select the polling or notification mode for presence events. Changes in the authorization of presence attributes are propagated to the watchers via the **notifySubscription()** operation, the blocking of a subscription by the presentity is propagated via a **subscriptionEnded** operation.

The sequence diagram does not show the internal communication within the presence server. It is assumed that the Presence Consumer and Supplier interfaces are implemented by the same instance. If an implementer of the API find other solutions preferable, he has to take care of the internal communication himself.

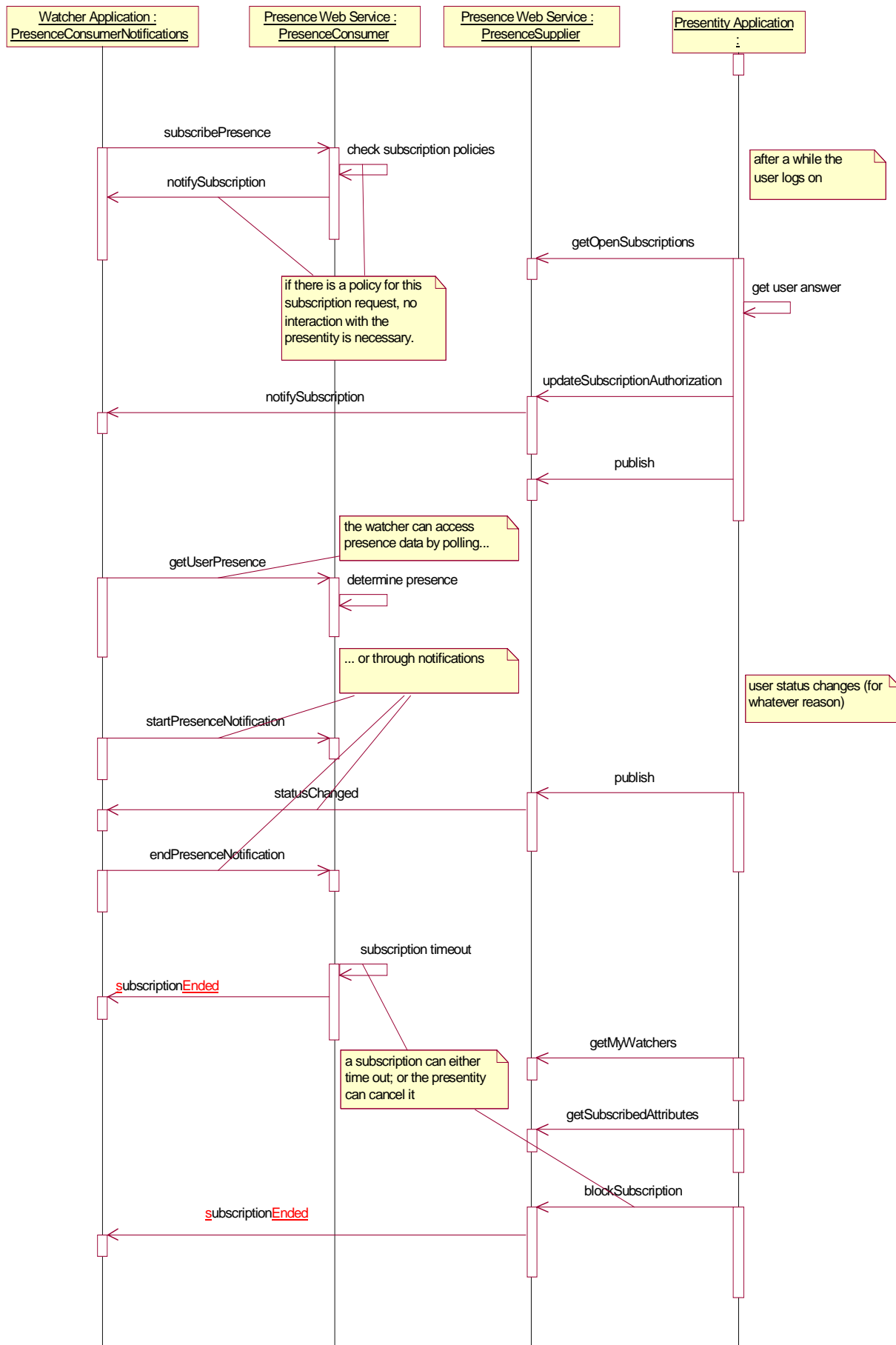


Figure 2: Message interaction overview

7 XML Schema data type definition

Presence attributes are inspired by the IETF's Rich Presence ideas (see Bibliography).

7.1 PresenceAttributeType enumeration

The different types of attributes. For each entry in this enumeration there is a separate value type.

| Enumeration value | Description |
|-------------------|--|
| Activity | The presentity's activity (available, busy, lunch, etc.) |
| Place | At what kind of place the presentity is (home, office, etc.) |
| Privacy | The amount of privacy the user wants (public, quiet, etc.) |
| Sphere | The user's current environment (work, home) |
| Communication | The user's means of communication (phone, mail, etc.) |
| Other | A name - value pair for arbitrary presence information |

7.2 ActivityValue enumeration

This enumeration shows the user's current activity. If the activity is unknown, the attribute value will be **ActivityNone**, meaning the attribute was not set. If the user is doing something not in this list, the value will be set to **ActivityOther**.

| Enumeration value | Description |
|-------------------|--|
| ActivityNone | Not set. |
| Available | The user is available for communication. |
| Busy | The user is busy and is only available for urgent matters. |
| DoNotDisturb | The user is very busy and does not wish to be disturbed. |
| OnThePhone | The user is on the phone. |
| Steering | The user is driving a car / train / airplane, etc. |
| Meeting | The user is in a meeting. |
| Away | No idea what the user is doing, but he is away. |
| Meal | The user is eating. |
| PermanentAbsence | The user is away and will not return for an extended period. |
| Holiday | The user is on holidays. |
| Performance | The user is in a theatre / concert. |
| InTransit | The user is in the transit area of an (air)port. |
| Travel | The user is travelling. |
| Sleeping | The user is sleeping. |
| ActivityOther | The user is doing something not in this list. |

7.3 PlaceValue enumeration

This enumeration shows the type of the user's current location. If the place type is unknown, the attribute value will be **PlaceNone**, meaning the attribute was not set. If the user is in a place not in this list, the value will be set to **PlaceOther**.

| Enumeration value | Description |
|-------------------|---|
| PlaceNone | Not set. |
| Home | The user is at home. |
| Office | The user is in an office. |
| PublicTransport | The user is on public transport. |
| Street | Walking on the street. |
| Outdoors | Generally outdoors. |
| PublicPlace | The user is in a public place. |
| Hotel | The user is in a hotel. |
| Theater | The user is in a theatre or concert. |
| Restaurant | The user is in a restaurant / bar / etc. |
| School | The user is at school. |
| Industrial | The user is in an industrial building. |
| Quiet | The user is in a quiet area. |
| Noisy | The user is in a noisy area. |
| Aircraft | The user is on an aircraft. |
| Ship | The user is on a ship. |
| Bus | The user is in a bus. |
| Station | The user is in a bus- or railway station. |
| Mall | The user is in a mall. |
| Airport | The user is in an airport. |
| Train | The user is in a train. |
| PlaceOther | The user is in a kind of place not listed here. |

7.4 PrivacyValue enumeration

This enumeration shows the amount of privacy a user currently has. If the privacy is unknown, the attribute value will be **PrivacyNone**, meaning the attribute was not set. If the privacy is not in this list, the value will be set to **PrivacyOther**.

| Enumeration value | Description |
|-------------------|---|
| PrivacyNone | Not set. |
| PrivacyPublic | The user is surrounded by other people and cannot discuss openly. |
| PrivacyPrivate | The user is alone and able to talk openly. |
| PrivacyQuiet | The user is in a quiet environment and cannot talk at all. |
| PrivacyOther | None of the other values applies. |

7.5 SphereValue enumeration

This enumeration shows the sphere within which the user acts. If the sphere is unknown, the attribute value will be **SphereNone**, meaning the attribute was not set. If the sphere is not in this list (neither work nor home), the value will be set to **SphereOther**.

| Enumeration value | Description |
|-------------------|---|
| SphereNone | Not set. |
| SphereWork | The user is acting within his work sphere, i.e. as a member of his company. |
| SphereHome | The user is acting within his home sphere, i.e. as a private person. |
| SphereOther | The user is acting neither within his work nor within his home sphere. |

7.6 CommunicationMeansType enumeration

This enumeration lists communication means. If the communication attribute refers to a means not in this list, it will point to **MeansOther**.

| Enumeration value | Description |
|-------------------|--|
| Phone | The communication attribute refers to a phone (fixed line or mobile or SIP). |
| Chat | The communication attribute refers to a chat client. |
| SMS | The communication attribute refers to an SMS client. |
| Video | The communication attribute refers to a video phone (fixed line or mobile or SIP). |
| Web | The communication attribute refers to a web client. |
| EEmail | The communication attribute refers to an e-mail client. |
| MMS | The communication attribute refers to an MMS client. |
| MeansOther | The communication attribute refers to any other client. |

7.7 CommunicationMeans structure

This structure describes on way of reaching the presentity.

| Element name | Element type | Optional | Description |
|--------------|------------------------|----------|---|
| priority | xsd:float | No | The priority of this communication means. Between 0 and 1, the latter meaning the highest priority. |
| contact | xsd:anyURI | No | The presentity's contact address for this communication means. |
| type | CommunicationMeansType | No | The type of this communication means. |

7.8 CommunicationValue structure

This structure describes the various ways of reaching a presentity.

| Element name | Element type | Optional | Description |
|--------------|--------------------------------------|----------|--|
| means | CommunicationMeans [0..unbounded] | Yes | The different ways of reaching the presentity. |

7.9 OtherValue structure

This structure can be used for storing arbitrary data about a presentity.

| Element name | Element type | Optional | Description |
|--------------|--------------|----------|-----------------------------|
| name | xsd:string | No | Description of the content. |
| value | xsd:string | No | Attribute content. |

7.10 PresenceAttribute structure

Presence data published by a presentity and retrieved by watchers.

| Element name | Element type | Optional | Description |
|--------------|--|----------|--|
| lastChange | xsd:dateTime | No | The time and date when the attribute was changed last. |
| note | xsd:string | Yes | An explanatory note. |
| type | PresenceAttributeType | No | Determines the type of the value field. |
| value | One of the six value types; depends on field "type" | No | The actual value of the attribute. |

This data structure is split into two types in the XSD file: A **PresenceAttribute** contains an **AttributeTypeAndValue**.

7.11 SubscriptionRequest structure

This structure is returned to the presentity by the Presence Web Service and contains the requesting watcher and the attributes he wants to subscribe.

| Element name | Element type | Optional | Description |
|--------------|--------------------------------------|----------|---|
| watcher | xsd:anyURI | No | The watcher who wants to gain access to data. |
| attributes | PresenceAttributeType [1..unbounded] | No | The attributes the watcher wants to see. |
| application | xsd:string | No | The name of the application running on behalf of the watcher. Note that this field has solely informative purposes, access rights management is based on watcher id only. |

7.12 PresencePermission structure

The answer from the service to the watcher in the **notifySubscription** operation.

| Element name | Element type | Optional | Description |
|--------------|-----------------------|----------|--|
| attribute | PresenceAttributeType | No | The name of the attribute the watcher wanted to subscribe. |
| decision | xsd:boolean | No | Whether the presentity accepted the subscription. If no, any further fields should be ignored. |

8 Web Service interface definition

This API is separated into three interfaces:

- PresenceConsumer interface: watcher methods for requesting and subscribing presence data.
- PresenceNotification interface: is the watcher notification interface for presence events.
- PresenceSupplier interface: presentity methods for supplying presence data and managing subscriptions.

8.1 Interface: PresenceConsumer

Client role: watcher.

This set of methods is used by the watcher to obtain presence data. After the subscription to presence data, the watcher can select between a polling mode or a notification mode in order to receive presence data.

8.1.1 Operation: subscribePresence

We assume that the watcher has been previously authenticated, so that his identity is known and can be associated with the subscription at the server.

The presentity is contacted and requested to authorize the watcher. As this process generally involves user interaction there cannot be an immediate response. The watcher is notified with **notifySubscription()**. If the presentity is a group, every member of the group will be contacted for authorization. The watcher will get one notification for each member.

Only after the subscription is completed (and the presentity has allowed access to attributes) may the watcher will get information when he uses **getUserPresence()** or **startPresenceNotification()**.

Note that the **SimpleReference** contains the correlator string used in subsequent messages to the notification interface.

At this interface level, the subscription has no expiration, although it can be ended from the presentity or the underlying layers (see **subscriptionEnded** operation).

8.1.1.1 Input message: subscribePresenceRequest

| Part name | Part type | Optional | Description |
|-------------|--------------------------------------|----------|--|
| presentity | xsd:anyURI | No | A presentity or a group of presentities whose attributes the watcher wants to monitor. |
| attributes | PresenceAttributeType [0..unbounded] | Yes | The attributes the watcher wants to access. (the same for all the group members). An empty array means subscription of all attributes. |
| application | xsd:string | No | Describes the application the watcher needs the data for. |
| reference | common:SimpleReference | No | The notification interface. |

8.1.1.2 Output message: subscribePresenceResponse

| Part name | Part type | Optional | Description |
|-----------|-----------|----------|-------------|
| None | | | |

8.1.1.3 Referenced faults

ServiceException from ES 202 391-1 [3]:

- SVC0001: Service error.
- SVC0002: Invalid input value.
- SVC0004: No valid addresses - if the presentity address does not exist.

PolicyException from ES 202 391-1 [3]:

- POL0006: Groups not allowed.
- POL0007: Nested groups not allowed.

8.1.2 Operation: getUserPresence

Returns the aggregated presence data of a presentity. Only the attributes which the watcher is entitled to see will be returned. This method does not support group identities.

Before getting these attributes, the watcher has to subscribe to them (see above). The presentity needs not be informed of the access, as he has already consented when the watcher called **subscribePresence()**.

8.1.2.1 Input message: getUserPresenceRequest

| Part name | Part type | Optional | Description |
|------------|--------------------------------------|----------|---|
| presentity | xsd:anyURI | No | The presentity whose data the watcher wants to see. |
| attributes | PresenceAttributeType [0..unbounded] | Yes | The attributes the watcher wants to see. An empty array means all attributes. |

8.1.2.2 Output message: getUserPresenceResponse

| Part name | Part type | Optional | Description |
|-----------|----------------------------------|----------|---------------------------|
| result | PresenceAttribute [0..unbounded] | Yes | The actual presence data. |

8.1.2.3 Referenced faults

ServiceException from ES 202 391-1 [3]:

- SVC0001: Service error.
- SVC0002: Invalid input value.
- SVC0004: No valid addresses - if the presentity address does not exist.

PolicyException from ES 202 391-1 [3]. The presentity has the possibility to cancel or block a subscription by manipulating the policy rules. The exception informs the watcher about this status change.

- POL0002: Privacy error - if the watcher is not subscribed to the requested data.
- POL0006: Groups not allowed.

8.1.3 Operation: startPresenceNotification

The notification pattern with correlation is used in order to be able to correlate the notification events with the request. The attributes represent a subset of the attributes subscribed and can be used as filter.

The watcher sets a notification trigger on certain user presence attribute changes. If the list of attributes is empty, the watcher wants to be notified on all subscribed attributes.

In case the presentity is a group the watcher will receive notifications for every single member of the group. The watcher will only get notifications for those attributes and presentities he subscribed successfully prior to the call. The service will return a list of presentities where the notifications could not be set up.

The presentity needs not be informed of the access, as he has already consented when the watcher called **subscribePresence()**.

Note that the **SimpleReference** contains the correlator string used in subsequent messages to the notification interface.

8.1.3.1 Input message: startPresenceNotificationRequest

| Part name | Part type | Optional | Description |
|----------------|--------------------------------------|----------|---|
| presentity | xsd:anyURI | No | The presentity or group whose attributes the watcher wants to monitor. |
| attributes | PresenceAttributeType [0..unbounded] | Yes | The attributes the watcher wants to see. |
| reference | common:SimpleReference | No | The notification interface. |
| frequency | common:TimeMetric | No | Maximum frequency of notifications (can also be considered minimum time between notifications). In case of a group subscription the service must make sure this frequency is not violated by notifications for various members of the group, especially in combination with checkImmediate . |
| duration | common:TimeMetric | Yes | Length of time notifications occur for, do not specify to use default notification time defined by service policy. |
| count | xsd:int | Yes | Maximum number of notifications. For no maximum, either do not specify this part or specify a value of zero. |
| checkImmediate | xsd:boolean | No | Whether to check status immediately after establishing notification. |

8.1.3.2 Output message: startPresenceNotificationResponse

| Part name | Part type | Optional | Description |
|-----------|---------------------------|----------|--|
| result | xsd:anyURI [0..unbounded] | Yes | The presentities whose attributes the watcher did not subscribe. Empty if all went fine. |

8.1.3.3 Referenced faults

ServiceException from ES 202 391-1 [3]:

- SVC0001: Service error.
- SVC0002: Invalid input value.
- SVC0004: No valid addresses - if the presentity URI does not exist.
- SVC0005: Duplicate correlator.

PolicyException from ES 202 391-1 [3]. The presentity has the possibility to cancel or block a subscription by manipulating the policy rules. The exception informs the watcher about this status change.

- POL0001: Policy error.
- POL0004: Unlimited notifications not supported.
- POL0005: Too many notifications requested.
- POL0006: Groups not allowed.
- POL0007: Nested groups not allowed.

8.1.4 Operation: endPresenceNotification

Indicates that the watcher does not want further notifications for a specific notification request (identified by the correlator). Note that the subscription to presence data stays active; the caller of this method remains a watcher and can still use **getUserPresence()** or reactivate the notifications.

8.1.4.1 Input message: endPresenceNotificationRequest

| Part name | Part type | Optional | Description |
|------------|------------|----------|---|
| correlator | xsd:string | No | The notification the watcher wants to cancel. |

8.1.4.2 Output message: endPresenceNotificationResponse

| Part name | Part type | Optional | Description |
|-----------|-----------|----------|-------------|
| None | | | |

8.1.4.3 Referenced faults

ServiceException from ES 202 391-1 [3]:

- SVC0001: Service error.
- SVC0002: Invalid input value.

PolicyException from ES 202 391-1 [3]:

- POL0001: Policy error.

8.2 Interface: PresenceNotification

This client callback interface is used by the presence consumer interface to send notifications.

8.2.1 Operation: statusChanged

The asynchronous operation is called by the Web Service when an attribute for which notifications were requested changes.

8.2.1.1 Input message: statusChangedRequest

| Part name | Part type | Optional | Description |
|-------------------|-------------------------------------|----------|---|
| correlator | xsd:string | No | Identifies the notification request. |
| presentity | xsd:anyURI | No | The presentity whose presence status has changed. |
| changedAttributes | PresenceAttribute [1..unbounded] | No | The new presence data. |

8.2.1.2 Output message: statusChangedResponse

| Part name | Part type | Optional | Description |
|-----------|-----------|----------|-------------|
| None | | | |

8.2.1.3 Referenced faults

None.

8.2.2 Operation: statusEnd

The notifications have ended for this correlator. This message will be delivered when the duration or count for notifications have been completed. This message will not be delivered in the case of an error ending the notifications or deliberate ending of the notifications (using **endPresenceNotification** operation).

8.2.2.1 Input message: statusEndRequest

| Part name | Part type | Optional | Description |
|------------|------------|----------|---|
| correlator | xsd:string | No | Correlator provided in request to set up this notification. |

8.2.2.2 Output message: statusEndResponse

| Part name | Part type | Optional | Description |
|-----------|-----------|----------|-------------|
| None | | | |

8.2.2.3 Referenced faults

None.

8.2.3 Operation: notifySubscription

This asynchronous method notifies the watcher that the server or the presentity handled the pending subscription.

8.2.3.1 Input message: notifySubscriptionRequest

| Part name | Part type | Optional | Description |
|------------|--------------------------------------|----------|---|
| presentity | xsd:anyURI | No | The presentity whose attributes the watcher wants to monitor. |
| decisions | PresencePermission [0..unbounded] | Yes | Denote the attributes the server/presentity accepted to expose. |

8.2.3.2 Output message: notifySubscriptionResponse

| Part name | Part type | Optional | Description |
|-----------|-----------|----------|-------------|
| None | | | |

8.2.3.3 Referenced faults

None.

8.2.4 Operation: subscriptionEnded

This asynchronous operation is called by the Web Service to notify the watcher (application) that the subscription has terminated. Typical reasons are a timeout of the underlying SIP soft state subscription (in accordance with [8] and [5]) or the decision of the presentity to block further presence information to that watcher. Since the subscription request has no expiration parameters, the service implementation may provide an inactivity timer that also triggers the **subscriptionEnded** operation.

8.2.4.1 Input message: subscriptionEndedRequest

| Part name | Part type | Optional | Description |
|------------|------------|----------|--|
| presentity | xsd:anyURI | No | The presentity to which the subscription has terminated. |
| reason | xsd:string | No | Timeout, Blocked. |

8.2.4.2 Output message: subscriptionEndedResponse

| Part name | Part type | Optional | Description |
|-----------|-----------|----------|-------------|
| None | | | |

8.2.4.3 Referenced faults

None.

8.3 Interface: PresenceSupplier

These methods are used by the presentity to supply presence data and manage access to the data by its watchers. We assume that the presentity has been previously authenticated, so that his Identity is known.

8.3.1 Operation: publish

The presentity publishes data about herself. This data will then be filtered by the system and forwarded to the watchers who have ordered notifications.

8.3.1.1 Input message: publishRequest

| Part name | Part type | Optional | Description |
|-----------|----------------------------------|----------|---|
| presence | PresenceAttribute [0..unbounded] | Yes | The presence attributes the devices of the presentity supports. |

8.3.1.2 Output message: publishResponse

| Part name | Part type | Optional | Description |
|-----------|-----------|----------|-------------|
| None | | | |

8.3.1.3 Referenced faults

ServiceException from ES 202 391-1 [3]:

- SVC0001: Service error.
- SVC0002: Invalid input value.

PolicyException from ES 202 391-1 [3]:

- POL0001: Policy error.

8.3.2 Operation: getOpenSubscriptions

Called periodically by the presentity to see if any watcher wants to subscribe to presence data. The client will answer open requests with **updateSubscriptionAuthorization()**.

8.3.2.1 Input message: getOpenSubscriptionsRequest

| Part name | Part type | Optional | Description |
|-----------|-----------|----------|-------------|
| None | | | |

8.3.2.2 Output message: getOpenSubscriptionsResponse

| Part name | Part type | Optional | Description |
|-----------|------------------------------------|----------|--------------------|
| result | SubscriptionRequest [0..unbounded] | Yes | Any open requests. |

8.3.2.3 Referenced faults

ServiceException from ES 202 391-1 [3]:

- SVC0001: Service error.

PolicyException from ES 202 391-1 [3]:

- POL0001: Policy error.

8.3.3 Operation: updateSubscriptionAuthorization

The presentity answers with this operation to watcher subscriptions for which no authorization policy exists. The answer consists of the attribute and the watcher involved and the permissions for each attribute. Subscription requests that are not answered are assumed pending.

The operation can be used by the presentity to change anytime the authorization for a certain watcher or group to monitor one or several attributes.

If the watcher did not try to subscribe the attribute - i.e. there is no pending subscription from this watcher to an attribute in the **decisions** array, a Presence-specific ServiceException (SVC0220) will be raised and the entire authorization request ignored.

8.3.3.1 Input message: updateSubscriptionAuthorizationRequest

| Part name | Part type | Optional | Description |
|-----------|-----------------------------------|----------|-------------------------------|
| watcher | xsd:anyURI | No | watcher or group of watchers. |
| decisions | PresencePermission [1..unbounded] | No | The answers to open requests. |

8.3.3.2 Output message: updateSubscriptionAuthorizationResponse

| Part name | Part type | Optional | Description |
|-----------|-----------|----------|-------------|
| None | | | |

8.3.3.3 Referenced faults

ServiceException from ES 202 391-1 [3]:

- SVC0001: Service error.
- SVC0002: Invalid input value.
- SVC0004: No valid addresses.
- SVC0220: No subscription request.

PolicyException from ES 202 391-1 [3]:

- POL0001: Policy error.

8.3.4 Operation: getMyWatchers

Returns an array of watching identities that are subscribed to the presentity's attributes. They are not necessarily users of the notification system, the mere fact that they are allowed to see the presentity's attributes is enough to be on this list.

8.3.4.1 Input message: getMyWatchersRequest

| Part name | Part type | Optional | Description |
|-----------|-----------|----------|-------------|
| None | | | |

8.3.4.2 Output message: getMyWatchersResponse

| Part name | Part type | Optional | Description |
|-----------|---------------------------|----------|--|
| result | xsd:anyURI [0..unbounded] | Yes | The list of identities who currently have access to the presentity's attributes. |

8.3.4.3 Referenced faults

ServiceException from ES 202 391-1 [3]:

- SVC0001: Service error.

PolicyException from ES 202 391-1 [3]:

- POL0001: Policy error.

8.3.5 Operation: getSubscribedAttributes

Returns an array of attributes that a specific watcher has subscribed.

8.3.5.1 Input message: getSubscribedAttributesRequest

| Part name | Part type | Optional | Description |
|-----------|------------|----------|---|
| watcher | xsd:anyURI | No | The watcher whose subscriptions the presentity wants to know. |

8.3.5.2 Output message: getSubscribedAttributesResponse

| Part name | Part type | Optional | Description |
|-----------|---|----------|--|
| result | PresenceAttributeType [0..unbounded] | Yes | The attributes the watcher is subscribed to. |

8.3.5.3 Referenced faults

ServiceException from ES 202 391-1 [3]:

- SVC0001: Service error.
- SVC0004: No valid addresses.
- SVC0221: Not a watcher - if the URI in the **watcher** part is not a watcher of the presentity.

PolicyException from ES 202 391-1 [3]:

- POL0001: Policy error.

8.3.6 Operation: blockSubscription

With this operation the presentity can block entirely the flow of presence information to a certain subscribed watcher by cancelling the subscription. The watcher will be notified with a **subscriptionEnded()** message.

8.3.6.1 Input message: blockSubscriptionRequest

| Part name | Part type | Optional | Description |
|-----------|------------|----------|---|
| watcher | xsd:anyURI | No | The watcher whose subscriptions the presentity wants to cancel. |

8.3.6.2 Output message: blockSubscriptionResponse

| Part name | Part type | Optional | Description |
|-----------|-----------|----------|-------------|
| None | | | |

8.3.6.3 Referenced faults

ServiceException from ES 202 391-1 [3]:

- SVC0001: Service error.
- SVC0002: Invalid input value.
- SVC0004: No valid addresses.
- SVC0221: Not a watcher - if the URI in the **watcher** part is not a watcher of the presentity.

PolicyException from ES 202 391-1 [3]:

- POL0001: Policy error.

9 Fault definitions

9.1 ServiceException

9.1.1 SVC0220: No subscription request

| Name | Description |
|-----------|---|
| messageId | SVC0220. |
| text | No subscription request from watcher %1 for attribute %2. |
| variables | %1 - watcher URI. %2 - type of attribute, from clause 7.1. |

9.1.2 SVC0221: Not a watcher

| Name | Description |
|-----------|----------------------|
| messageId | SVC0221. |
| text | %1 is not a watcher. |
| variables | %1 - watcher URI. |

10 Service policies

Service policies for this service.

| Name | Type | Description |
|------------------------------|-------------------|---|
| MaximumNotificationFrequency | common:TimeMetric | Maximum rate of notification delivery (also can be considered minimum time between notifications). |
| MaximumNotificationDuration | common:TimeMetric | Maximum amount of time a notification may be set up for. |
| DefaultNotificationDuration | common:TimeMetric | Default amount of time for which a notification will be set up. |
| MaximumCount | xsd:int | Maximum number of notifications that may be requested. |
| UnlimitedCountAllowed | xsd:boolean | Allowed to specify unlimited notification count (i.e. either by not specifying the optional count part in the startPresenceNotificationRequest message or by specifying a value of zero). |
| GroupSupport | xsd:boolean | Groups may be included with addresses. |
| NestedGroupSupport | xsd:boolean | Are nested groups supported in group definitions. |

Annex A (normative): WSDL for Presence

The document/literal WSDL representation of this interface specification is compliant to ES 202 391-1 [3] and is contained in text files (contained in archive es_20239114v010201p0.zip) which accompany the present document.

Annex B (informative): Bibliography

3GPP: "IETF Dependencies and Priorities". http://www.3gpp.org/tb/Other/IETF_archive_07.03.05/IETF.htm

IETF RFC 4660: "Functional Description of Event Notification Filtering". <http://www.ietf.org/rfc/rfc4660.txt>.

IETF RFC 4480: "RPID: Rich Presence: Extensions to the Presence Information Data Format (PIDF)".
<http://www.ietf.org/rfc/rfc4480.txt>.

draft-ietf-simple-xcap-11: "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)".
<http://www.ietf.org/internet-drafts/draft-ietf-simple-xcap-11.txt>.

NOTE: This version expires November 6, 2006. Also reference item #57 in 3GPP: "IETF Dependencies and Priorities".

ETSI TS 123 141: "Universal Mobile Telecommunications System (UMTS); Presence service; Architecture and functional description; Stage 2 (3GPP TS 23.141)".

Repository of information about the Extensible Messaging and Presence Protocol (XMPP), which was contributed by the Jabber Software Foundation (JSF) to the IETF, <http://www.xmpp.org/>.

History

| Document history | | |
|-------------------------|---------------|--|
| V1.1.1 | March 2005 | Publication |
| V1.2.1 | October 2006 | Membership Approval Procedure MV 20061215: 2006-10-17 to 2006-12-15 |
| V1.2.1 | December 2006 | Publication |
| | | |
| | | |