

**Open Service Access (OSA);  
Application Programming Interface (API);  
Part 14: Presence and Availability Management SCF  
(Parlay 5)**



---

Reference

DES/TISPAN-01005-14-OSA

---

Keywords

API, IDL, OSA, UML

**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](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 2005.

© The Parlay Group 2005.

All rights reserved.

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

# Contents

Intellectual Property Rights .....	9
Foreword.....	9
1 Scope .....	10
2 References .....	10
3 Definitions and abbreviations.....	10
3.1 Definitions .....	10
3.2 Abbreviations .....	10
4 Presence and Availability Management SCF .....	11
4.1 Introduction .....	11
4.2 Motivation .....	11
4.3 Goals .....	11
4.4 Concepts .....	12
4.4.1 Identity .....	12
4.4.2 Agent .....	12
4.4.3 Presence .....	13
4.4.4 Availability .....	14
4.4.5 Events .....	15
4.5 Scope of PAM information .....	15
4.6 Security and privacy .....	15
5 Sequence Diagrams .....	16
5.1 Use of authentication tokens .....	16
5.2 Event registration and notification .....	17
6 Class Diagrams .....	18
6.1 PAM Provisioning SCF Class Diagrams .....	18
6.2 PAM Access SCF Class Diagrams .....	20
6.3 PAM Event SCF Class Diagrams .....	21
7 The Service Interface Specifications .....	21
7.1 Interface Specification Format .....	21
7.1.1 Interface Class .....	22
7.1.2 Method descriptions.....	22
7.1.3 Parameter descriptions .....	22
7.1.4 State Model.....	22
7.2 Base Interface .....	22
7.2.1 Interface Class IpInterface .....	22
7.3 Service Interfaces .....	22
7.3.1 Overview .....	22
7.4 Generic Service Interface .....	23
7.4.1 Interface Class IpService .....	23
7.4.1.1 Method setCallback() .....	23
7.4.1.2 Method setCallbackWithSessionID().....	23
8 Presence and Availability Management Interface Classes .....	24
8.1 PAM Provisioning SCF Interface Classes.....	24
8.1.1 Interface Class IpPAMProvisioningManager .....	24
8.1.1.1 Method getAuthToken().....	25
8.1.1.2 Method obtainInterface() .....	25
8.1.1.3 Method <<new>> getAccessControl().....	25
8.1.1.4 Method <<new>> setAccessControl() .....	26
8.1.2 Interface Class IpPAMIdentityManagement .....	26
8.1.2.1 Method createIdentity().....	27
8.1.2.2 Method deleteIdentity().....	28
8.1.2.3 Method isIdentity().....	28

8.1.2.4	Method createGroupIdentity().....	28
8.1.2.5	Method deleteGroupIdentity().....	29
8.1.2.6	Method addToGroup() .....	29
8.1.2.7	Method removeFromGroup() .....	30
8.1.2.8	Method listMembers().....	30
8.1.2.9	Method isGroupIdentity() .....	30
8.1.2.10	Method listGroupMembership() .....	31
8.1.2.11	Method addAlias() .....	31
8.1.2.12	Method removeAliases() .....	31
8.1.2.13	Method listAliases() .....	32
8.1.2.14	Method lookupByAlias() .....	32
8.1.2.15	Method associateTypes() .....	33
8.1.2.16	Method disassociateTypes().....	33
8.1.2.17	Method listTypesOfIdentity() .....	33
8.1.2.18	Method hasType() .....	34
8.1.2.19	Method getIdentityAttributes() .....	34
8.1.2.20	Method setIdentityAttributes().....	35
8.1.3	Interface Class IpPAMAgentManagement .....	35
8.1.3.1	Method createAgent() .....	36
8.1.3.2	Method deleteAgent() .....	37
8.1.3.3	Method isAgent() .....	37
8.1.3.4	Method enableCapabilities() .....	37
8.1.3.5	Method disableCapabilities() .....	38
8.1.3.6	Method listEnabledCapabilities().....	38
8.1.3.7	Method listAllCapabilities().....	38
8.1.3.8	Method isCapableOf().....	39
8.1.3.9	Method associateTypes() .....	39
8.1.3.10	Method disassociateTypes().....	40
8.1.3.11	Method listTypesOfAgent() .....	40
8.1.3.12	Method hasType() .....	40
8.1.3.13	Method getAgentAttributes() .....	41
8.1.3.14	Method setAgentAttributes().....	41
8.1.4	Interface Class IpPAMAgentAssignment .....	42
8.1.4.1	Method assignAgent() .....	42
8.1.4.2	Method unassignAgent() .....	43
8.1.4.3	Method listAssignedAgents().....	43
8.1.4.4	Method listAssociatedIdentitiesOfAgent().....	43
8.1.4.5	Method listAssignedAgentsByCapability() .....	44
8.1.4.6	Method listCapabilitiesOfIdentity() .....	44
8.1.4.7	Method isIdentityCapableOf() .....	45
8.1.5	Interface Class IpPAMIdentityTypeManagement .....	45
8.1.5.1	Method createIdentityAttribute() .....	46
8.1.5.2	Method deleteIdentityAttribute() .....	46
8.1.5.3	Method getIdentityAttributeDefinition().....	47
8.1.5.4	Method listAllIdentityAttributes() .....	47
8.1.5.5	Method createIdentityType().....	47
8.1.5.6	Method deleteIdentityType().....	48
8.1.5.7	Method listIdentityTypes().....	48
8.1.5.8	Method addIdentityTypeAttributes() .....	48
8.1.5.9	Method removeIdentityTypeAttributes() .....	49
8.1.5.10	Method listIdentityTypeAttributes() .....	49
8.1.6	Interface Class IpPAMAgentTypeManagement .....	49
8.1.6.1	Method createAgentAttribute() .....	50
8.1.6.2	Method deleteAgentAttribute() .....	50
8.1.6.3	Method getAgentAttributeDefinition() .....	51
8.1.6.4	Method listAllAgentAttributes() .....	51
8.1.6.5	Method createAgentType() .....	51
8.1.6.6	Method deleteAgentType() .....	52
8.1.6.7	Method listAgentTypes() .....	52
8.1.6.8	Method addAgentTypeAttributes() .....	52
8.1.6.9	Method removeAgentTypeAttributes().....	53
8.1.6.10	Method listAgentTypeAttributes().....	53

8.1.7	Interface Class IpPAMCapabilityManagement .....	53
8.1.7.1	Method createCapabilityAttribute() .....	54
8.1.7.2	Method deleteCapabilityAttribute() .....	55
8.1.7.3	Method getCapabilityAttributeDefinition() .....	55
8.1.7.4	Method listAllCapabilityAttributes() .....	55
8.1.7.5	Method createCapability() .....	56
8.1.7.6	Method deleteCapability() .....	56
8.1.7.7	Method listCapabilities().....	56
8.1.7.8	Method addCapabilityAttributes() .....	57
8.1.7.9	Method removeCapabilityAttributes() .....	57
8.1.7.10	Method listCapabilityAttributes() .....	57
8.1.7.11	Method assignCapabilitiesToType() .....	58
8.1.7.12	Method unassignCapabilitiesFromType().....	58
8.1.7.13	Method listCapabilitiesOfType() .....	58
8.2	PAM Access SCF Interface Classes .....	59
8.2.1	Interface Class IpPAMPresenceAvailabilityManager .....	59
8.2.1.1	Method getAuthToken().....	59
8.2.1.2	Method obtainInterface() .....	60
8.2.1.3	Method <<new>> getAccessControl().....	60
8.2.1.4	Method <<new>> setAccessControl() .....	61
8.2.1.5	Method <<new>> activateService().....	61
8.2.1.6	Method <<new>> deactivateService() .....	61
8.2.1.7	Method <<new>> isActiveIdentity() .....	62
8.2.2	Interface Class IpPAMIdentityPresence .....	62
8.2.2.1	Method setIdentityPresence().....	63
8.2.2.2	Method setIdentityPresenceExpiration().....	63
8.2.2.3	Method getIdentityPresence() .....	63
8.2.3	Interface Class IpPAMAvailability.....	64
8.2.3.1	Method getAvailability().....	65
8.2.3.2	Method getPreference().....	66
8.2.3.3	Method setPreference() .....	66
8.2.4	Interface Class IpPAMAgentPresence.....	67
8.2.4.1	Method setAgentPresence() .....	67
8.2.4.2	Method setCapabilityPresence() .....	68
8.2.4.3	Method setAgentPresenceExpiration().....	68
8.2.4.4	Method setCapabilityPresenceExpiration().....	69
8.2.4.5	Method getAgentPresence().....	69
8.2.4.6	Method getCapabilityPresence() .....	70
8.2.5	Interface Class IpAppPAMPreferenceCheck.....	70
8.2.5.1	Method computeAvailability() .....	71
8.3	PAM Event SCF Interface Classes .....	71
8.3.1	Interface Class IpPAMEventManager .....	71
8.3.1.1	Method getAuthToken().....	72
8.3.1.2	Method obtainInterface() .....	72
8.3.1.3	Method <<new>> getAccessControl().....	73
8.3.1.4	Method <<new>> setAccessControl() .....	73
8.3.1.5	Method <<new>> activateService().....	74
8.3.1.6	Method <<new>> deactivateService() .....	74
8.3.1.7	Method <<new>> isActiveIdentity() .....	74
8.3.2	Interface Class IpPAMEventHandler.....	75
8.3.2.1	Method isRegistered().....	75
8.3.2.2	Method registerAppInterface().....	76
8.3.2.3	Method registerForEvent().....	76
8.3.2.4	Method deregisterAppInterface().....	77
8.3.2.5	Method deregisterFromEvent() .....	77
8.3.3	Interface Class IpAppPAMEventHandler.....	77
8.3.3.1	Method eventNotify().....	78
8.3.3.2	Method eventNotifyErr() .....	78
9	State Transition Diagrams .....	78
10	PAM Service Properties .....	78

10.1	PAM Access Service .....	79
10.2	PAM Event Service .....	79
11	PAM Data Definitions .....	79
11.1	Entity Address Definitions .....	79
11.1.1	TpPAMFQName.....	79
11.1.2	TpPAMFQNameList .....	79
11.2	Attribute Data Definitions .....	79
11.2.1	TpPAMAttribute .....	79
11.2.2	TpPAMAttributeList.....	80
11.2.3	TpPAMAttributeDef .....	80
11.2.4	TpPAMAttributeDefList.....	80
11.3	Presence Data Definitions .....	80
11.3.1	TpPAMCapability.....	80
11.3.2	TpPAMCapabilityList .....	80
11.3.3	TpPAMPresenceData.....	81
11.3.4	TpPAMPresenceDataList .....	81
11.4	Pre-defined Presence type .....	81
11.4.1	Preentity .....	81
11.5	Availability Data Definitions .....	81
11.5.1	TpPAMAvailabilityProfile .....	81
11.5.2	TpPAMAvailabilityProfileList .....	81
11.5.3	TpPAMPrivacyCode.....	81
11.6	Availability Context Data Definitions .....	82
11.6.1	TpPAMContext.....	82
11.6.2	TpPAMContextName .....	82
11.6.3	TpPAMContextData .....	82
11.6.4	TpPAMCommunicationContext .....	82
11.6.5	TpPAMContextList .....	82
11.7	Credential data definitions.....	83
11.7.1	TpPAMCredential.....	83
11.8	Availability and Access Control Preference Data Definitions .....	83
11.8.1	IpAppPAMPreferenceCheckRef.....	83
11.8.2	TpPAMAccessControlData .....	83
11.8.3	TpPAMACLDefault .....	83
11.8.4	TpPAMPreferenceOp .....	83
11.8.5	TpPAMPreferenceType .....	84
11.8.6	TpPAMPreferenceData.....	84
11.9	Time data definitions .....	84
11.9.1	TpPAMTimeInterval .....	84
11.10	Pre-defined Entity Types and Attributes .....	84
11.11	Interface name definitions .....	85
11.11.1	TpPAMProvisioningInterfaceName .....	85
11.11.2	TpPAMPresenceAvailabilityInterfaceName.....	85
11.11.3	TpPAMEventInterfaceName .....	85
11.12	Event data definitions .....	86
11.12.1	IpAppPAMEventHandlerRef.....	86
11.12.2	TpPAMClientID .....	86
11.12.3	TpPAMEventID .....	86
11.12.4	TpPAMEventName .....	86
11.12.5	TpPAMEventNameList .....	86
11.12.6	TpPAMEventInfo .....	87
11.12.7	TpPAMEventInfoList .....	87
11.12.8	TpPAMNotificationInfo .....	87
11.12.9	TpPAMNotificationInfoList .....	87
11.12.10	PAM_CE_IDENTITY_CREATED.....	88
11.12.10.1	TpPAMICEEventData.....	88
11.12.10.2	TpPAMICNotificationData.....	88
11.12.11	PAM_CE_IDENTITY_DELETED .....	88
11.12.11.1	TpPAMIDEEventData.....	88
11.12.11.2	TpPAMIDNotificationData.....	88
11.12.12	PAM_CE_GROUP_MEMBERSHIP_CHANGED .....	89

11.12.12.1	TpPAMGMCEventData.....	89
11.12.12.2	TpPAMGMNotificationData.....	89
11.12.13	PAM_CE_AGENT_CREATED.....	89
11.12.13.1	TpPAMACEventData.....	89
11.12.13.2	TpPAMACNotificationData.....	89
11.12.14	PAM_CE_AGENT_DELETED.....	90
11.12.14.1	TpPAMADEventData.....	90
11.12.14.2	TpPAMADNotificationData.....	90
11.12.15	PAM_CE_AGENT_ASSIGNED.....	90
11.12.15.1	TpPAMAAEventData.....	90
11.12.15.2	TpPAMAANotificationData.....	91
11.12.16	PAM_CE_AGENT_UNASSIGNED.....	91
11.12.16.1	TpPAMAUEventData.....	91
11.12.16.2	TpPAMAUNotificationData.....	91
11.12.17	PAM_CE_CAPABILITY_CHANGED.....	91
11.12.17.1	TpPAMCCEventData.....	92
11.12.17.2	TpPAMCCNotificationData.....	92
11.12.18	PAM_CE_AGENT_CAPABILITY_PRESENCE_SET.....	92
11.12.18.1	TpPAMACPSEventData.....	92
11.12.18.2	TpPAMACPSNotificationData.....	93
11.12.19	PAM_CE_AGENT_PRESENCE_SET.....	93
11.12.19.1	TpPAMAPSEventData.....	93
11.12.19.2	TpPAMAPSNotificationData.....	93
11.12.20	PAM_CE_IDENTITY_PRESENCE_SET.....	93
11.12.20.1	TpPAMIPSEventData.....	94
11.12.20.2	TpPAMIPSNotificationData.....	94
11.12.21	PAM_CE_AVAILABILITY_CHANGED.....	94
11.12.21.1	TpPAMAVCEventData.....	95
11.12.21.2	TpPAMAVCNotificationData.....	95
11.12.22	PAM_CE_WATCHERS_CHANGED.....	95
11.12.22.1	TpPAMWCEventData.....	96
11.12.22.2	TpPAMWCNotificationData.....	96
11.12.22.3	TpPAMwatcherChangeType.....	96
11.13	Error Types.....	97
11.13.1	TpPAMErrorCause.....	97
11.13.2	TpPAMErrorInfo.....	97
12	Presence and Availability Management Exception Classes.....	97
<b>Annex A (informative): Further PAM Information.....</b>		<b>99</b>
A.1	UML Models.....	99
A.1.1	Identity.....	99
A.1.2	Agent.....	100
A.2	Model.....	100
A.3	Architecture.....	101
A.4	Levels of access.....	102
A.4.1	Application.....	103
A.4.2	Service.....	103
A.4.3	Thin client.....	103
A.5	Use cases.....	104
A.5.1	Identity Management.....	104
A.5.2	Agent Management.....	104
A.5.3	Agent Assignment.....	105
A.5.4	Agent Presence.....	105
A.5.5	Identity Presence.....	105
A.5.6	Availability.....	105
<b>Annex B (normative): OMG IDL Description of Presence and Availability Management SCF.....</b>		<b>106</b>

<b>Annex C (informative):</b>	<b>Java™ API Description of the Presence and Availability Management SCFs.....</b>	<b>107</b>
<b>Annex D (informative):</b>	<b>Contents of 3GPP OSA R6 Presence and Availability Management.....</b>	<b>108</b>
<b>Annex E (informative):</b>	<b>Description of Presence and Availability Management for 3GPP2 cdma2000 networks .....</b>	<b>109</b>
E.1	General Exceptions.....	109
E.2	Specific Exceptions .....	109
E.2.1	Clause 1: Scope .....	109
E.2.2	Clause 2: References .....	109
E.2.3	Clause 3: Definitions and abbreviations .....	109
E.2.4	Clause 4: Presence and Availability Management SCF .....	109
E.2.5	Clause 5: Sequence Diagrams .....	109
E.2.6	Clause 6 Class Diagrams.....	109
E.2.7	Clause 7: The Service Interface Specifications .....	109
E.2.8	Clause 8: Presence and Availability Management Interface Classes .....	110
E.2.9	Clause 9: State Transition Diagrams .....	110
E.2.10	Clause 10: PAM Service Properties .....	110
E.2.11	Clause 11: PAM Data Definitions.....	110
E.2.12	Clause 12: Presence and Availability Management Exception Classes .....	110
E.2.13	Annex B (normative): OMG IDL Description of Presence and Availability Management SCF .....	110
E.2.14	Annex C (informative): Java™ API Description of Presence and Availability Management SCF.....	110
<b>Annex F (informative):</b>	<b>Record of changes .....</b>	<b>111</b>
F.1	Interfaces .....	111
F.1.1	New .....	111
F.1.2	Deprecated.....	111
F.1.3	Modified.....	111
F.1.4	Removed.....	111
F.2	Methods.....	112
F.2.1	New .....	112
F.2.2	Deprecated.....	112
F.2.3	Modified.....	112
F.2.4	Removed.....	112
F.3	Data Definitions .....	112
F.3.1	New .....	112
F.3.2	Modified.....	112
F.3.3	Removed.....	113
F.4	Service Properties.....	113
F.4.1	New .....	113
F.4.2	Deprecated.....	113
F.4.3	Modified.....	113
F.4.4	Removed.....	113
F.5	Exceptions .....	113
F.5.1	New .....	113
F.5.2	Modified.....	114
F.5.3	Removed.....	114
F.6	Others .....	114
History .....		115



---

# 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), and is now submitted for the ETSI standards Membership Approval Procedure.

The present document is part 14 of a multi-part deliverable covering Open Service Access (OSA); Application Programming Interface (API), as identified below. The API specification (ES 203 915) is structured in the following parts:

- Part 1: "Overview";
- Part 2: "Common Data Definitions";
- Part 3: "Framework";
- Part 4: "Call Control";
- Part 5: "User Interaction SCF";
- Part 6: "Mobility SCF";
- Part 7: "Terminal Capabilities SCF";
- Part 8: "Data Session Control SCF";
- Part 9: "Generic Messaging SCF";
- Part 10: "Connectivity Manager SCF";
- Part 11: "Account Management SCF";
- Part 12: "Charging SCF";
- Part 13: "Policy Management SCF";
- Part 14: "Presence and Availability Management SCF";**
- Part 15: "Multi-media Messaging SCF".

The present document has been defined jointly between ETSI, The Parlay Group (<http://www.parlay.org>) and the 3GPP, in co-operation with a number of JAIN™ Community (<http://www.java.sun.com/products/jain>) member companies.

**The present document forms part of the Parlay 5.0 set of specifications.**

**A subset of the present document is in 3GPP TS 29.198-14 V6.2.0 (Release 6).**

---

# 1 Scope

The present document is part 14 of the Stage 3 specification for an Application Programming Interface (API) for Open Service Access (OSA).

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

The present document specifies the Presence and Availability Management Service Capability Feature (SCF) aspects of the interface. All aspects of the Presence and Availability Management SCF are defined here, these being:

- Sequence Diagrams.
- Class Diagrams.
- Interface specification plus detailed method descriptions.
- State Transition diagrams.
- Data Definitions.
- IDL Description of the interfaces.

The process by which this task is accomplished is through the use of object modelling techniques described by the Unified Modelling Language (UML).

---

# 2 References

The references listed in clause 2 of ES 203 915-1 contain provisions which, through reference in this text, constitute provisions of the present document.

ETSI ES 203 915-1: "Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview (Parlay 5)".

ETSI ES 203 915-2: "Open Service Access (OSA); Application Programming Interface (API); Part 2: Common Data Definitions (Parlay 5)".

---

# 3 Definitions and abbreviations

## 3.1 Definitions

For the purposes of the present document, the terms and definitions given in ES 203 915-1 apply.

## 3.2 Abbreviations

For the purposes of the present document, the abbreviations defined in ES 203 915-1 apply.

---

## 4 Presence and Availability Management SCF

### 4.1 Introduction

The goal of these interfaces is to establish a standard for maintaining, retrieving and publishing information about:

- Digital identities;
- Characteristics and presence status of agents (representing capabilities for communication, content delivery, etc.);
- Capabilities and state of entities; and
- Presence and Availability of entities for various forms of communication and the contexts in which they are available.

Establishing such a standard in the industry will facilitate creation of many inter-operable services over multiple network technologies and, in addition, allow end users greater flexibility in managing their services and communication capabilities while addressing their privacy concerns.

### 4.2 Motivation

Consider the following simple but desirable scenario for a communication service: An end-user wishes to receive instant messages from her management at any time on her mobile phone, from co-workers only on her desktop computer, and in certain cases for the messages to be forwarded to e-mail or even a fax machine/printer. The senders may know her availability for various forms of communication in the way she chooses to reveal it or alternatively the senders may never know how she will be receiving their messages. This scenario spans over multiple services and protocols and can only be solved currently by a proprietary solution that maintains the required information in an ad-hoc fashion within the application.

PAM is not a replacement for the protocols being standardized for various communication and network services. PAM attempts to standardize the management and sharing of presence and availability information across multiple services and networks.

The PAM specification is motivated by the observations that:

- The notions of Identity, Presence and Availability are common to but independent of the various communication technologies, protocols and applications that provide services using these technologies.
- Presence does not necessarily imply availability. End-users or organizations require greater control over making themselves available through various communication devices.
- Presence based services need to address privacy concerns on who can access presence information and under what conditions.

Management of availability will span over multiple communication services and service providers.

### 4.3 Goals

The main goal of Presence and Availability Management is to facilitate the development of a rich set of applications and services that span over multiple communication systems (instant messaging, e-mail, fax, telephony, etc.) and to provide the end user greater flexibility and control in managing their communications. A standardized platform allows software developers to create communication management applications that are independent of the underlying technologies and protocols.

As the next step in the evolution of directory and database enabled applications and services, separation of the management of identities and availability of users or organizations from specific applications enables uniform and centralized administration of data and creates the potential to bring control over communication services to the user's desktops.

The purpose of the present document is to adopt the first release of a Presence and Availability Management interface specification created by an industry consortium, PAMforum, established for this purpose harmonized with the IETF model for presence (RFC 2778). The present document is also consistent with the ongoing work in 3GPP for defining the requirements and architecture for a standard presence service in the network.

With a desired goal of rapid acceptance and usage, the specification has been deliberately designed to be as simple as possible with an attempt to include a minimal set of functionality that is sufficient for use in non-trivial applications. Often, this has been at the cost of some useful features, which would have made the specification baroque and cumbersome if not controversial.

## 4.4 Concepts

This clause briefly describes the various concepts involved in the present document to serve as the context for the rest of the document.

### 4.4.1 Identity

Identity, for purposes of the PAM specification, is a limited electronic representation of an entity (i.e. an individual or an organization) that participates in PAM-enabled applications and services. This concept corresponds to the concept of Presentity as described in the IETF Common Presence and Instant Messaging Model (RFC 2778).

The main characteristic of an entity that is central to PAM specifications is the name (or handle) by which entities are identified by applications and services. Entities may have multiple names, login ids, account names, etc., by which they are identified. As PAM attempts to abstract over multiple networks and services, it does not assume that a single name will necessarily identify entities across all application domains.

The generalized structure available in 3GPP for user names that may contain various formats for addressing has been adopted for these specifications.

To enable entities to be identified by any of the names associated with them, PAM identities can be assigned aliases. A name and a namespace pair can be defined as an alias of another name and namespace pair. It is important to note that aliases are just synonyms and hence have limited semantics. In particular, they are not powerful enough to model personas each with their own capabilities and privacy requirements.

An identity can represent a single entity or a group of identities. Group identities have similar semantics to non-group identities but, in addition, maintain a list of identities that constitute the group. As an example, a sales department may be modelled as a group identity with the identities of the members of the department being member identities of the group. Group identities and their member identities do not inherit anything from each other.

No other relationships between identities are within the scope of the PAM specifications.

For flexibility and extensibility, attribute lists are used to associate additional data with identities. Identities are typed to provide a way to manage such attribute lists. An identity type may be associated with a specific set of attributes and all identities of that type inherit instances of such attributes.

For consistency with IETF (RFC 2778) defined presence data models, PAM pre-defines an identity type Presentity with a list of presence attributes based on the definitions in RFC 2778.

PAM implementations may map certain existing directory and database data to one or more types to allow access via PAM interfaces. PAM specifications do not specify how the data within the profiles are to be stored. They may be stored within the PAM implementation or mapped to data stored on external directories and databases.

### 4.4.2 Agent

An agent, for PAM purposes, is a limited electronic representation of a software or hardware device through which identities manifest themselves or make themselves available to applications and services.

An important characteristic of an agent is a list of one or more capabilities associated with it. A capability is what makes an agent useful. A capability either represents the ability of an agent to participate in communications and content delivery (e.g.: instant messaging, SMS, WAP, voice) or it represents the ability of an agent to report useful information (e.g. location, velocity, temperature, mood) of the environment around it.

PAM does not specify any pre-defined capabilities. Applications may define and use their own capabilities.

Agent instances are identified by names (or handles). As for identities, names exist in the context of a namespace. Within a namespace, a name is assumed to be unique. Two agent instances can have the same name as long as they are in different namespaces. For example, a mobile phone and a PDA manufactured by two different manufacturers may coincidentally have the same serial number by which they are identified. As PAM attempts to unify services over multiple technologies, it does not assume that a name uniquely identifies agent instances across all technologies or across all manufacturers. They can be disambiguated through the use of namespaces.

No relationships between agents are within the scope of the PAM specifications.

For flexibility and extensibility, attribute lists are used to associate additional data with agents. Agents are typed to provide a way to manage such attribute lists. An agent type may be associated with a specific set of attributes and all agents of that type inherit instances of such attributes.

PAM does not specify any pre-defined attributes or types. Applications may define and use their own agent types.

PAM implementations may map certain existing directory and database data to one or more types to allow access via PAM interfaces. PAM specifications do not specify how the data within the profiles are to be stored. They may be stored within the PAM implementation or mapped to data stored on external directories and databases.

Agent instances are associated with one or more identities. This association results in the inheritance of associated agents' capabilities by the identities.

### 4.4.3 Presence

The concept of presence has been used in several application areas, being most explicit in Instant Messaging. Starting from a simple notion of online/offline status, it has expanded to include other context information around the status such as disposition (out to lunch, away from the computer, etc.) and activity status (on the phone, idle, etc.). Location information, on the other hand, has largely been kept separate from what has been traditionally considered presence information. PAM specifications broaden the concepts of presence recognizing that all such information, including location, describes different contexts of an entity's existence. The unifying property is that the presence information is continually changing and that there is value in knowing the current information at different points in time for services and applications.

For the purposes of PAM specifications, presence is an extensible set of characteristics that captures the dynamic context in which an identity or an agent exists at any point in time. In contrast to the relatively static information about identities or agents (e.g. names, addresses, capabilities), presence refers to dynamic information such as location, status, disposition, etc. Registrations of presence and location information in existing applications are covered by this definition.

Presence information is differentiated from the more static information associated with identities and agents that are stored in attributes. The rationalization for this design is that the presence information is dynamic and has implications on the implementation. Some of the presence information is too dynamic to be maintained in static data stores such as directories and without this hint about the data characteristics, PAM implementers may make sub-optimal decisions on the way the data is stored. Second, presence information typically has expiration data that needs to be understood by the implementation.

The PAM specification recognizes that devices that provide presence information are not necessarily devices that communicate. Certain agents may report presence information but not be capable of communication. Certain agents may be communication devices but may not be able to provide presence information. In general, the presence of an identity is computed from presence information provided by one or more agents and the ability to communicate is derived from one or more communication-capable agents available to the identity.

The PAM specification does not specify the methods by which the presence information is derived. An agent may explicitly register its own presence information or the information may be derived from other network elements. For example, an instant messaging client on a desktop computer can register its status based on when a user is logged in. A mobile phone may do an explicit registration on a WAP server for instant messaging. The phone's presence for voice calls, on the other hand, may be inferred implicitly by querying the cellular network for the device being on when requested. The presence of an identity, on the other hand, may be computed using presence information from one or more devices owned by the identity.

Finally, the PAM specification does not require that the presence information be stored explicitly (i.e. in a materialized fashion) in a PAM implementation. An implementation may infer the presence information on demand from the underlying services or networks.

For compatibility with the presence model from IETF (RFC 2778), a type called Presentity is pre-defined with the attributes consistent with the IETF Presence model.

#### 4.4.4 Availability

Availability is a property of an identity denoting its ability and willingness to share information about itself or to communicate with another identity based on factors such as the type of communication requested, the identity of the calling entity and the preferences and policies that are associated with the recipient. This is the primary means by which the current PAM specification enables controls for privacy. While presence is, in most applications, a necessity for availability, presence does not necessarily imply availability to all.

Availability is always with respect to a context. A context in PAM specifications is a set of attributes defining the state in which the availability is requested. For example, the query "Is Jane available for IM for Rob?" identifies the type of communication and the identity of the asker as the context. PAM allows for availability to be differentiated based on any attribute of a context. A context, "Communication" is pre-defined in PAM.

Most queries for presence in existing applications can be mapped into PAM availability queries to control the information being given out. Alternatively, queries can be mapped directly into PAM presence queries in situations where privacy controls and policies are not required or all presence data is open to the entity querying. This allows PAM specifications to be consistent with existing presence servers and to serve as the basis for presence services across multiple protocols while providing uniform and flexible privacy controls.

PAM specification does not specify whether the availability is computed on demand or stored explicitly. In some applications, the availability may be pre-computed and stored explicitly while in some, it may be computed at each request for availability.

While the PAM specification provides a mechanism to associate preferences with an Identity to control availability, it neither specifies the syntax and semantics of the preferences nor the process by which the availability is computed. These aspects are left to the implementation.

For example, a particular implementation may provide the facility to store preferences as rules such as "I prefer to receive my instant messages on my computer rather than my cell phone unless the message is from my boss or the computer is off, etc."

As an example, a computation of availability for communication may consist of the following algorithm:

- 1) Find all devices of the identity being called that are capable of the specified form of communication AND have registered their presence status as available.
- 2) Evaluate the rules associated with the identity being called to select the preferred device(s) from the set of present devices determined in Step 1.
- 3) If there are any devices available satisfying Step 2, indicate the availability of the identity being called via the available devices.

An implementation can chose to provide one or more means to specify preferences. It is expected that if there is industry standardization on the specification of preferences, the implementations will support such a standard. This is currently outside the scope of PAM.

## 4.4.5 Events

Events are representations of certain identified occurrences related to the concepts described above. The PAM specification provides for registering interest (i.e. callbacks) in being notified of such occurrences. Any entity that subscribes to the Event is a "watcher" in the IETF terminology (RFC 2778). An implementation is expected to provide such notifications.

Examples of events include:

- Creation/deletion of an identity.
- Association of an agent instance with an identity.
- Change in presence status or location of an agent instance.
- Change in the presence information of an identity.
- Change in availability of an identity for a particular form of communication.

PAM specifications contain a set of pre-defined events. Each event is defined by a name of the event, a set of input attribute value pairs that must be provided when an event is registered for and a set of attribute value pairs that are included in the notifications sent out when the event of interest occurs.

## 4.5 Scope of PAM information

Presence and Availability Management has the following types of information in its scope:

- Identities, which consist of names and aliases of entities participating in communications.
- Agent information, which consists of names and communication capabilities of software and/or hardware devices.
- Agent provisioning, which consists of associations between instances of agents and identities.
- Presence information, which consists of an identity's or an agent's dynamic characteristics such as status and geographical location.
- Availability information, which consists of preferences associated with identities and computation of availability, based on the devices present and the current preferences.
- Notification of changes to the above pieces of information.
- Security issues for access to this information.

The PAM specification consists of interfaces to manage or access the above information.

The specification purposefully does not include:

- Storage design or storage requirements for any of the presence and availability information.

These are to be decided by specific implementations of the PAM specification.

## 4.6 Security and privacy

As the Presence and Availability Management interface is designed to share information across administrative domains and to facilitate availability computation based on the identity of the entity desiring communication, security and privacy issues are addressed in the design. Two of the issues considered to be within the scope of PAM are:

- Access control to an implementation of the PAM specification.
- Use of an authenticated entity's credentials by methods in the specification.

To understand the distinction between the first two issues, consider, for example, an end-user that logs on to an Instant Messaging client and wishes to send a message. The client (or a gateway to which the client talks to) may access a PAM implementation to determine the availability of the destination for the message. The client (or the gateway) will need to be authorized for access to the PAM implementation independent of the user that logs in. A gateway may, in fact, do this access on behalf of a number of clients and, for performance reasons, wish to authenticate itself just once on start up rather than at each invocation. This authentication is handled by the authentication mechanisms in the OSA Framework common to all services within OSA.

Second, each invocation of a particular method will need to contain the credentials of the end-user that logged into the client so that the computation of the availability can take that into account when necessary for privacy issues.

It should be noted that the PAM specification allows for the possibility that the authentication of the end-users is not necessarily done within the PAM implementation itself. As long as the authenticated credentials supplied by the client (or gateway) are acceptable for validation and the client (or the gateway) itself is authenticated by the implementation, the authentication of end-users can occur anywhere outside the PAM implementation. A deployment scenario for a particular application is that one or more authentication services are provided as external services over PAM implementations.

This design does not preclude the possibility that the client (or the gateway) cannot be authenticated. Therefore, the credentials supplied by the client (or the gateway) may be held to stronger authentication criterion than credentials supplied by a trusted client (or gateway).

Finally, the PAM specification does not mandate the use of authentication within an implementation if the environment in which it is used does not require it.

Clause 5.1 explains the mechanism for providing data about the asker to each of the methods with a sequence diagram.

Privacy issues are addressed primarily by providing a mechanism to control the information flowing out of a PAM implementation based on whatever criterion the end user may choose to specify in the availability preferences and independent of any particular application.

The following security issues were considered to be outside the scope of PAM:

- Authentication of the identity of the end-users or entities. As explained above, this authentication may be provided by a third-party authentication service or it may occur through an authentication service written over the PAM platform. The only requirement is that the type of credentials supplied by the authentication service be acceptable to the PAM platform implementation being accessed.
- Encryption of the flow of information between a PAM platform implementation and clients of this implementation. This is dependent on the method of access to the interface which is outside the scope of the PAM specification and hence to be determined by the implementation.

---

## 5 Sequence Diagrams

Most of the methods in the PAM interfaces are independently used to query or update presence and availability related information with no transactions or state transitions involved. There are two use cases for which sequence diagrams are useful:

- Acquiring and using authentication tokens.
- Registering for PAM events and getting notifications on the occurrence of the event.

The sequence diagrams for these two cases are provided below. It is assumed that the authentication with the OSA framework has already occurred and the application has access to the PAM interfaces.

### 5.1 Use of authentication tokens

As an OSA Service, PAM uses the authentication features of the OSA Framework to provide access control to the PAM interfaces. In addition, PAM provides an optional mechanism for service/application level identification and authentication of the entity requesting the operation or alternatively on whose behalf the operation is being requested.

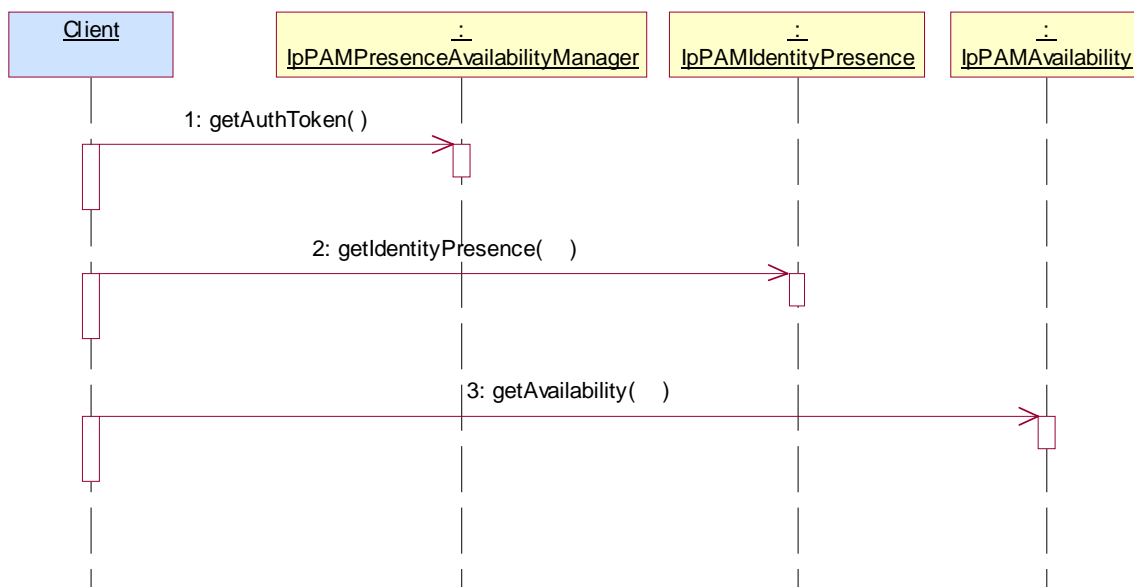


To handle privacy requirements, the results of presence and availability data updates or queries are dependent on the entity requesting the operations.

In the simplest case, the entity authenticating to the OSA Framework to get access to the service interface is the entity requesting the operation. In general, however, a proxy or an application (such as a messaging server or a conferencing server) may authenticate with the OSA Framework once and then check for presence and availability on behalf of multiple client applications (such as instant messaging clients). The credential of these client applications if and when needed by the PAM service can then be provided via the credential parameter in each of the interface methods.

The mechanism to provide the asker data is via the optional parameter of type TpPAMCredential in each of the methods. Supplying the entire asker data in each of the methods is expensive for an implementation since it will need to parse and validate the data supplied in the asker data structure each time. An application may be accessing multiple methods for itself or for the benefit of end user(s) and will need to supply the relevant asker data in each case. To make the consideration of asker data more efficient, the application uses the `getAuthToken()` method in each of the managers in the SCFs once for each session per asker and gets a credential that can be reused as many times as necessary in the same session to represent the same asker.

The sequence diagram for an example usage is given below.



1: For any unique entity requesting the operation, the authenticated client of the OSA PAM service, requests for an authentication token using the `getAuthToken()` method in the PresenceAvailability Manager interface.

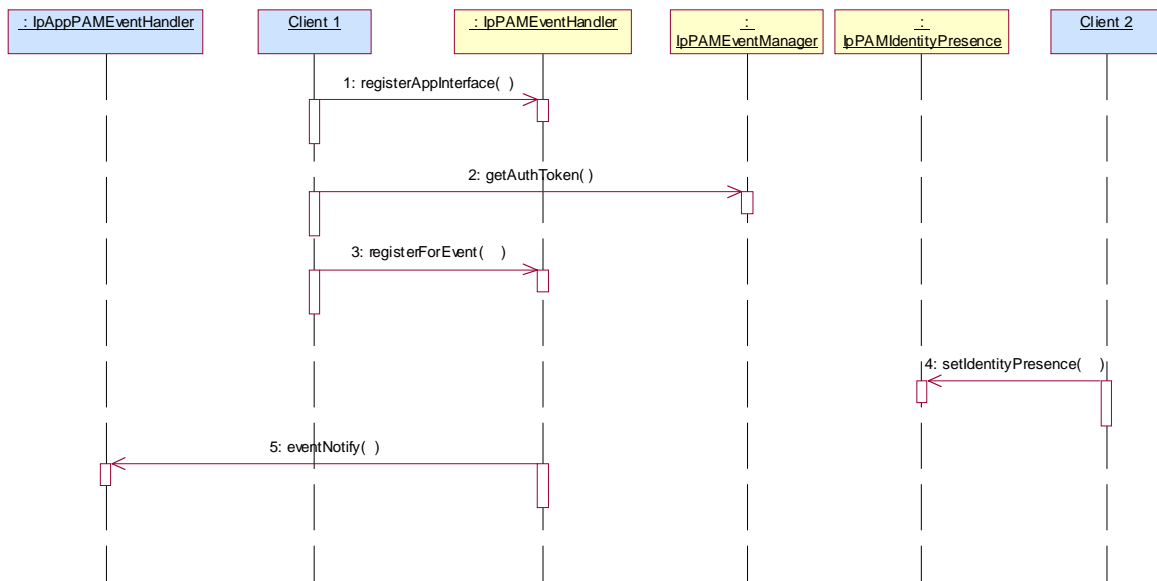
2: The token returned by the `getAuthToken()` method is used as the credential parameter of the `getIdentityPresence()` request.

3: The same token is used as the credential parameter of the `getAvailability()` request(). An authorization token can be used multiple times within the same session established with OSA framework.

## 5.2 Event registration and notification

An OSA client can register for certain events in the PAM service either for itself or on behalf of its own clients. The client will register one or more application interfaces with the event management service and then activate one or more events for each such registered interface.

The sequence diagram for an example is given below.



1: The client uses the registerAppInterface() method to register its notification interface. The getAuthToken() can be null since the client is doing this registration on its own behalf. The client gets a unique client ID back.

2: The client uses the getAuthToken() to get authentication credentials for its own application client on behalf of whom an event registration is required.

3: The client uses the registerForEvent() method to register for a change in availability event on behalf of its own client. The client gets a unique event ID back.

4: The presence information for an identity of interest in 3 above is changed by another client application acting on its behalf using the setIdentityPresence() method.

5: When the change in presence results in a change in the availability of the identity for the client that has registered for the availability change, a notification is sent out using the previously registered application interface.

## 6 Class Diagrams

PAM consists of the following SCFs:

- PAM Provisioning Service consisting of Interfaces to provision identities, agents and relationships between them;
- PAM Access Service consisting of interfaces to view and update presence and availability information; and
- PAM EventManagement Service consisting of interfaces to subscribe to events in PAM and be notified of such events.

### 6.1 PAM Provisioning SCF Class Diagrams

PAM Provisioning Service contains the following service interfaces. It contains no application interfaces.

#### IpPAMIdentityManagement

- The purpose of this interface is to manage subscriber names, aliases, groups and sets of attributes called Profiles associated with subscribers.

#### IpPAMAgentManagement

- The purpose of this interface is to manage agent (that models a hardware or software device) names, communication capabilities and sets of attributes called Profiles associated with agents.

## IpPAMAgentAssignment

- The purpose of this interface is to manage the relationship between identities and the agents assigned to them.

## IpPAMIdentityTypeManagement

## IpPAMAgentTypeManagement

## IpPAMCapabilityManagement

- These three interfaces allow for definitions of identity types, agent types and agent capabilities used for presence and availability management.

The interfaces and the relationships between them are shown in figure 1.

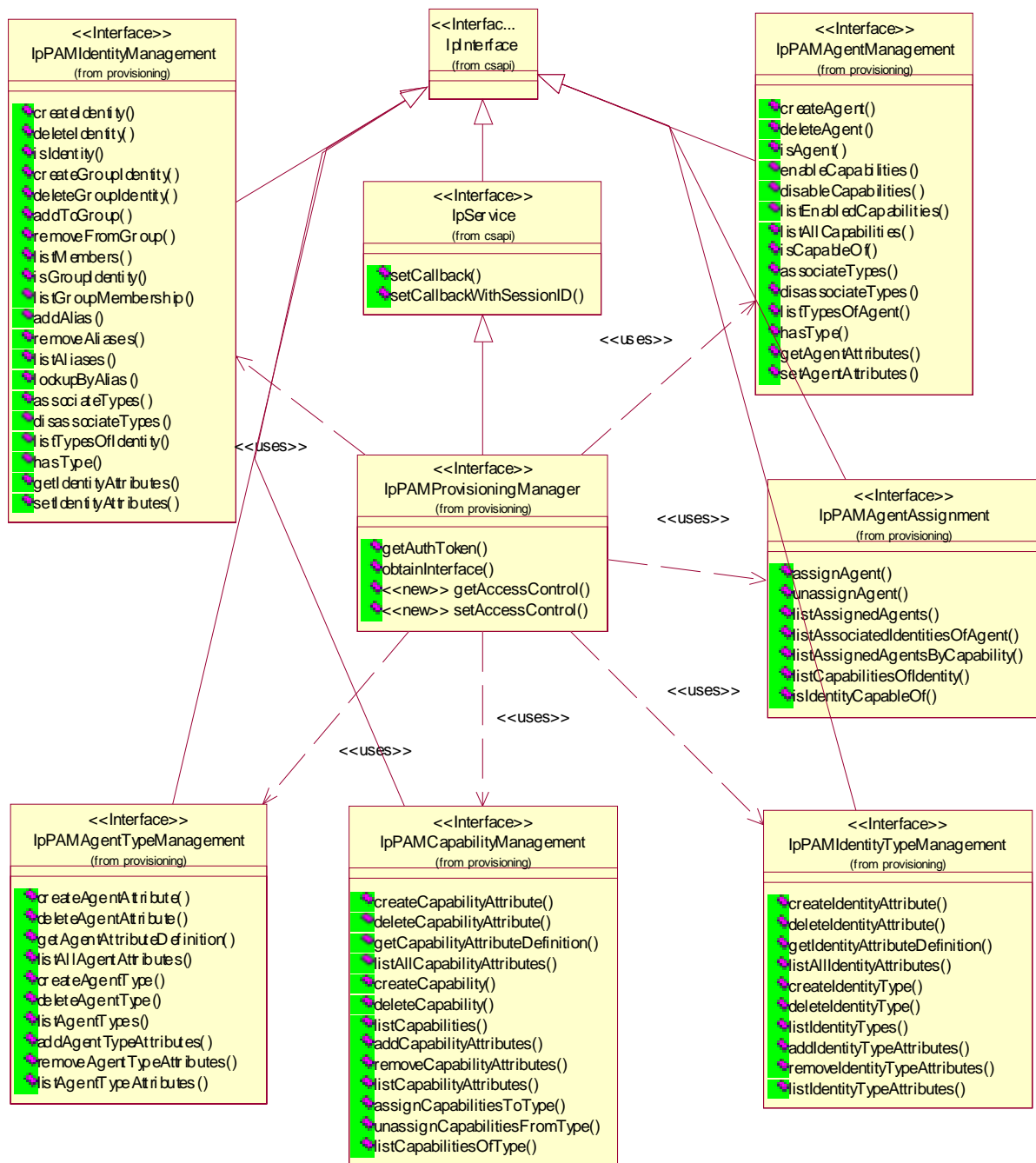


Figure 1: PAM Provisioning Service

## 6.2 PAM Access SCF Class Diagrams

The PAM Access service consists of two packages, one for the application interfaces and one for the service interfaces. The application PAM Access package consists of 0 or more instances of the IpAppPAMPreferenceCheck interface and the PAM event management service package consists of a single instance of the following interfaces obtainable by applications using the service interface IpPAMPresenceAvailabilityManager.

### IpPAMAgentPresence

- The purpose of this interface is to maintain the dynamic presence information of agents.

### IpPAMIdentityPresence

- The purpose of this interface is to maintain the dynamic presence information of identities.

### IpPAMAvailabilityManagement

- The purpose of this interface is to (i) Manage the preferences specified for the availability of an identity (ii) Query for the availability of identities for specific capabilities.

The interfaces and the relationships between them are shown in figure 2.

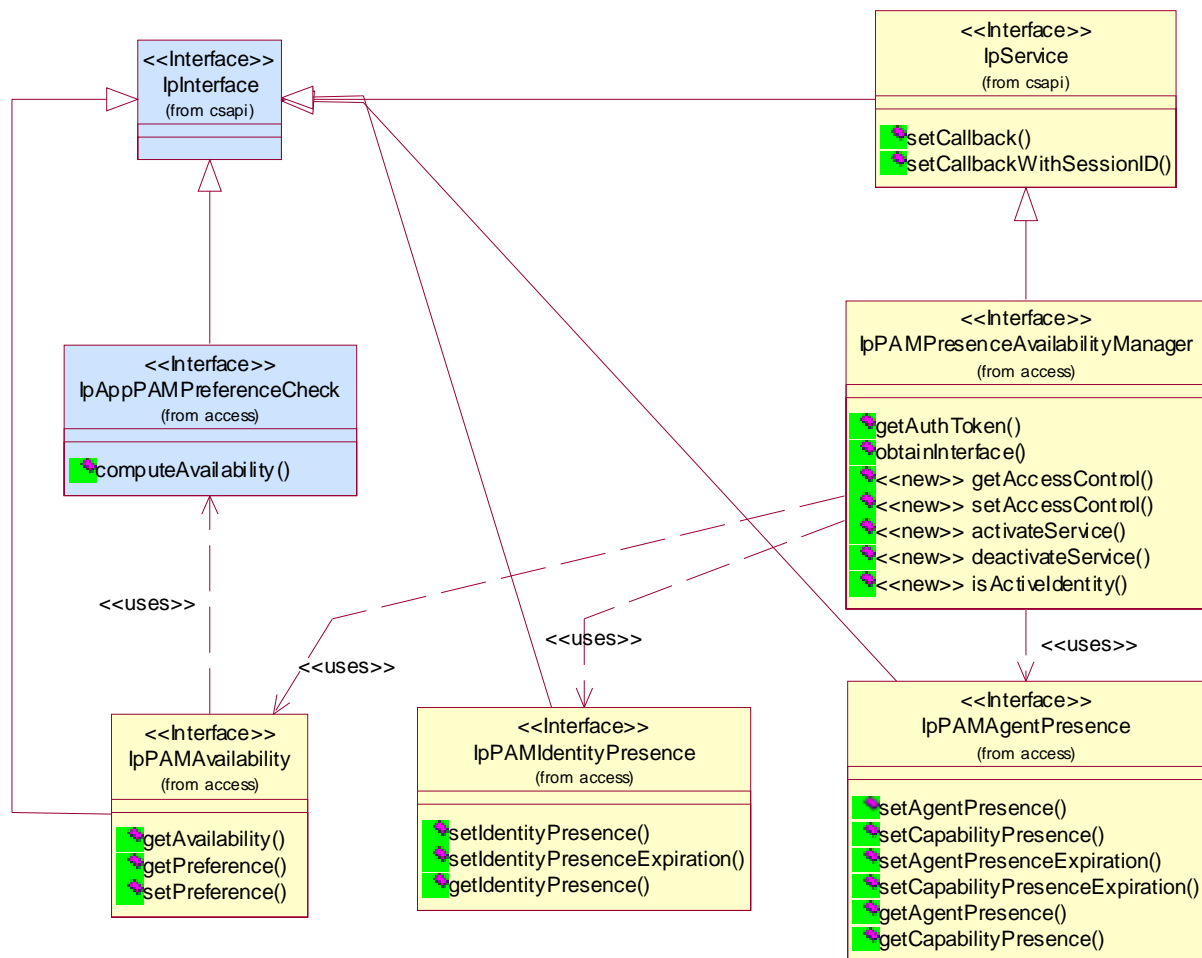


Figure 2: PAM Access Service

### 6.3 PAM Event SCF Class Diagrams

The PAM Event Management service consists of two packages, one for the application interfaces and one for the service interfaces. The application PAM event management package consists of 0 or more instances of the IpAppPAMEventHandler interface and the PAM event management service package consists of a single instance of the IpPAMEventHandler interface. This interface can be obtained by application using the service interface IpPAMEventManager.

Figure 3 shows the interfaces of PAM Event Management and the relationships between them.

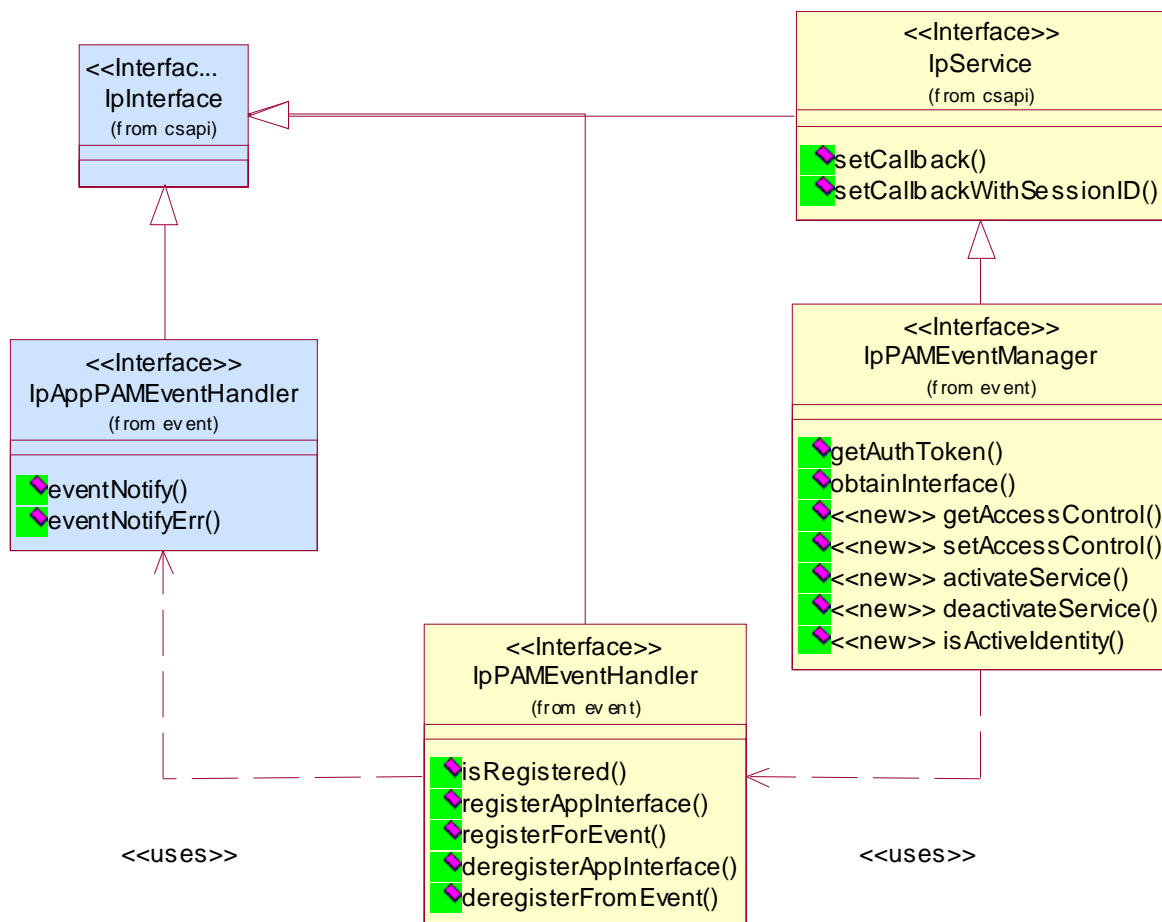


Figure 3: PAM Event Management Service

---

## 7 The Service Interface Specifications

### 7.1 Interface Specification Format

This clause defines the interfaces, methods and parameters that form a part of the API specification. The Unified Modelling Language (UML) is used to specify the interface classes. The general format of an interface specification is described below.

### 7.1.1 Interface Class

This shows a UML interface class description of the methods supported by that interface, and the relevant parameters and types. The Service and Framework interfaces for enterprise-based client applications are denoted by classes with name `Ip<name>`. The callback interfaces to the applications are denoted by classes with name `IpApp<name>`. For the interfaces between a Service and the Framework, the Service interfaces are typically denoted by classes with name `IpSvc<name>`, while the Framework interfaces are denoted by classes with name `IpFw<name>`.

### 7.1.2 Method descriptions

Each method (API method 'call') is described. Both synchronous and asynchronous methods are used in the API. Asynchronous methods are identified by a 'Req' suffix for a method request, and, if applicable, are served by asynchronous methods identified by either a 'Res' or 'Err' suffix for method results and errors, respectively. To handle responses and reports, the application or service developer must implement the relevant `IpApp<name>` or `IpSvc<name>` interfaces to provide the callback mechanism.

### 7.1.3 Parameter descriptions

Each method parameter and its possible values are described. Parameters described as 'in' represent those that must have a value when the method is called. Those described as 'out' are those that contain the return result of the method when the method returns.

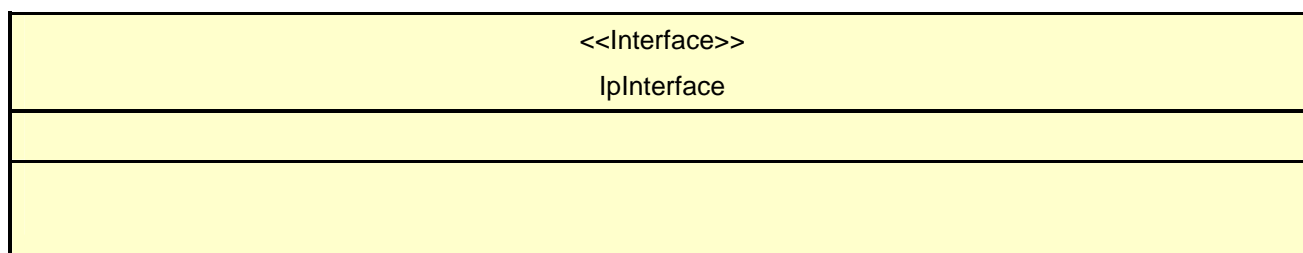
### 7.1.4 State Model

If relevant, a state model is shown to illustrate the states of the objects that implement the described interface.

## 7.2 Base Interface

### 7.2.1 Interface Class IpInterface

All application, framework and service interfaces inherit from the following interface. This API Base Interface does not provide any additional methods.



## 7.3 Service Interfaces

### 7.3.1 Overview

The Service Interfaces provide the interfaces into the capabilities of the underlying network - such as call control, user interaction, messaging, mobility and connectivity management.

The interfaces that are implemented by the services are denoted as 'Service Interface'. The corresponding interfaces that must be implemented by the application (e.g. for API callbacks) are denoted as 'Application Interface'.

## 7.4 Generic Service Interface

### 7.4.1 Interface Class IpService

Inherits from: IpInterface;

All service interfaces inherit from the following interface.

<<Interface>> IpService
<pre> setCallback (appInterface : in IpInterfaceRef) : void setCallbackWithSessionID (appInterface : in IpInterfaceRef, sessionID : in TpSessionID) : void </pre>

#### 7.4.1.1 Method setCallback()

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

##### *Parameters*

**appInterface : in IpInterfaceRef**

Specifies a reference to the application interface, which is used for callbacks.

##### *Raises*

**TpCommonExceptions, P\_INVALID\_INTERFACE\_TYPE**

#### 7.4.1.2 Method setCallbackWithSessionID()

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not use SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

##### *Parameters*

**appInterface : in IpInterfaceRef**

Specifies a reference to the application interface, which is used for callbacks.

**sessionID : in TpSessionID**

Specifies the session for which the service can invoke the application's callback interface.

##### *Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_INVALID\_INTERFACE\_TYPE**

## 8 Presence and Availability Management Interface Classes

PAM consists of the following SCFs:

- PAM Provisioning Service.
- PAM Access Service.
- PAM Event Service.

The PAM Provisioning service consists of the Identity Management, Agent Management, Agent Assignment, Agent Type Management, Identity Type Management and Capability Management interfaces.

The PAM Access service consists of the agent presence, identity presence and availability interfaces. The agent presence interface is not obtainable in the 3GPP release of the service.

The Event service consists of the Event Management interfaces.

An implementation of this API which supports or implements a method described in the present document, shall support or implement the functionality described for that method, for at least one valid set of values for the parameters of that method. Where a method is not supported by an implementation of a Service interface, the exception `P_METHOD_NOT_SUPPORTED` shall be returned to any call of that method.

### 8.1 PAM Provisioning SCF Interface Classes

This service consists of interfaces to provision identities, agents and the relationships between them.

#### 8.1.1 Interface Class IpPAMProvisioningManager

Inherits from: IpService.

The purpose of this interface is to supply the various interfaces available in this service to the application and to provide the authentication credentials. This interface is the only discoverable interface from the framework.

All PAM methods use an authentication token as a parameter since the outcome of the operations may depend on the entity requesting the operation. To enable this, the `getAuthToken()` method is used to obtain an implementation dependent token. An application that has authenticated itself with the OSA framework, can get an authentication token for itself. Alternatively, if the application is requesting PAM operations on behalf of multiple entities, authentication tokens may be requested for each such entity after providing any available data about the asker. These tokens can then be used repeatedly for operations within a session without further need to identify the asker.

<<Interface>> IpPAMProvisioningManager
<pre> getAuthToken (askerData : in TpAttributeList) : TpPAMCredential obtainInterface (interfaceName : in TpPAMProvisioningInterfaceName) : IpInterfaceRef &lt;&lt;new&gt;&gt; getAccessControl (identity : in TpPAMFQName, authToken : in TpPAMCredential) :   TpPAMAccessControlData &lt;&lt;new&gt;&gt; setAccessControl (identity : in TpPAMFQName, operation : in TpPAMPreferenceOp,   newAccessControl : in TpPAMAccessControlData, authToken : in TpPAMCredential) : void           </pre>



### 8.1.1.1 Method getAuthToken()

Get an authentication token for access to the interface methods.

Returns an implementation-dependent authentication credential that can be verified.

#### *Parameters*

**askerData : in TpAttributeList**

Specifies information about the asker. Can be an empty array. The exact attributes in this list are dependent on the application. PAM reserves the attribute "name" with type TpPAMFQName to contain the identity of the asker if known.

#### *Returns*

**TpPAMCredential**

#### *Raises*

**TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.1.2 Method obtainInterface()

Obtain available interfaces from the service. The valid parameters for this method can be obtained from the service property P\_OBTAINABLE\_INTERFACES.

Returns the requested interface.

#### *Parameters*

**interfaceName : in TpPAMProvisioningInterfaceName**

Specifies the name of the required interface.

#### *Returns*

**IpInterfaceRef**

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNAVAILABLE\_INTERFACE**

### 8.1.1.3 Method <<new>> getAccessControl()

Get the access control associated with the data belonging to an identity. The data associated with an identity includes the static and dynamic attributes of an identity as well as data about agents associated with an identity.

This method should be used in conjunction with the setAccessControl method.

Returns the access control if previously specified for the identity. Is null if there is no access control associated.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity of interest.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpPAMAccessControlData***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL****8.1.1.4 Method <<new>> setAccessControl()**

Set the access controls for the data associated with the specified identity. If the identity is Null, the access control is set for all identities (if authorized to do so). The data associated with an identity includes the static and dynamic attributes of an identity as well as data about agents associated with an identity.

Any existing access control will be modified based on the operation.

If the new access control is specified as Null for replace operation , an existing access control will be removed.

*Parameters***identity : in TpPAMFQName**

Specifies the identity of interest.

**operation : in TpPAMPreferenceOp**

Specifies the operation to be performed with the specified preference.

**newAccessControl : in TpPAMAccessControlData**

Specifies the access controls to add.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL****8.1.2 Interface Class IpPAMIdentityManagement**

Inherits from: IpInterface.

The purpose of this interface is to manage end-user or entity names, aliases, groups and sets of attributes associated with identities. An implementation may map these methods to operations on existing directories or databases. Some implementations may choose to provide a read-only access to the identity information.

The names of identities within a namespace must be unique. Each implementation exports an identifier as the default namespace that it serves. The identity name and the namespace may be used as an alias to another identity in a different namespace.

Aliases are associated with a given identity or group identity. Aliases must be uniquely assigned. In other words, two identities may not share the same alias.

This interface is meant for use by provisioning applications that establish and maintain identity names.

<<Interface>> IpPAMIdentityManagement
<pre> createlIdentity (identity : in TpPAMFQName, identityTypes : in TpStringList, authToken : in   TpPAMCredential) : void deletelIdentity (identity : in TpPAMFQName, authToken : in TpPAMCredential) : void isIdentity (identity : in TpPAMFQName, authToken : in TpPAMCredential) : TpBoolean createGroupIdentity (identity : in TpPAMFQName, identityTypes : in TpStringList, authToken : in   TpPAMCredential) : void deleteGroupIdentity (identity : in TpPAMFQName, authToken : in TpPAMCredential) : void addToGroup (group : in TpPAMFQName, member : in TpPAMFQName, authToken : in TpPAMCredential) :   void removeFromGroup (group : in TpPAMFQName, identity : in TpPAMFQName, authToken : in   TpPAMCredential) : void listMembers (identity : in TpPAMFQName, authToken : in TpPAMCredential) : TpPAMFQNameList isGroupIdentity (identity : in TpPAMFQName, authToken : in TpPAMCredential) : TpBoolean listGroupMembership (identity : in TpPAMFQName, authToken : in TpPAMCredential) : TpPAMFQNameList addAlias (identity : in TpPAMFQName, alias : in TpPAMFQName, authToken : in TpPAMCredential) : void removeAliases (identity : in TpPAMFQName, alias : in TpPAMFQName, authToken : in TpPAMCredential) :   void listAliases (identity : in TpPAMFQName, authToken : in TpPAMCredential) : TpPAMFQNameList lookupByAlias (alias : in TpPAMFQName, authToken : in TpPAMCredential) : TpPAMFQName associateTypes (identity : in TpPAMFQName, identityTypes : in TpStringList, authToken : in   TpPAMCredential) : void disassociateTypes (identity : in TpPAMFQName, identityTypes : in TpStringList, authToken : in   TpPAMCredential) : void listTypesOfIdentity (identity : in TpPAMFQName, authToken : in TpPAMCredential) : TpPAMFQNameList hasType (identity : in TpPAMFQName, typeName : in TpString, authToken : in TpPAMCredential) :   TpBoolean getIdentityAttributes (identity : in TpPAMFQName, identityType : in TpString, attributeNames : in   TpStringList, authToken : in TpPAMCredential) : TpPAMAttributeList setIdentityAttributes (identity : in TpPAMFQName, identityType : in TpString, attributes : in   TpPAMAttributeList, authToken : in TpPAMCredential) : void </pre>

### 8.1.2.1 Method createlIdentity()

Create a new non-Group Identity with the specified name. Names must be unique across both group identities and non-group identities. Names must be unique across types within a namespace.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the Identity to be created.

**identityTypes : in TpStringList**

Specifies the identity's associated types. Can be an empty array.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_IDENTITY\_EXISTS, P\_PAM\_UNKNOWN\_TYPE,  
P\_PAM\_INVALID\_CREDENTIAL**

#### 8.1.2.2 Method deleteIdentity()

Delete the specified identity and all its related data. Upon successful completion, associated aliases and attribute instances are deleted from the system. The identity is also removed from all groups of which the identity is a member.

*Parameters*

**identity : in TpPAMFQName**

Specifies the Identity to be deleted.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

#### 8.1.2.3 Method isIdentity()

Check if the specified non-group Identity exists. The method returns false for group identities.

Returns true if an identity with the specified name exists and false otherwise.

*Parameters*

**identity : in TpPAMFQName**

Specifies the Identity to be checked.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns*

**TpBoolean**

*Raises*

**TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL**

#### 8.1.2.4 Method createGroupIdentity()

Create a new Group Identity with the specified name. Name must be unique across both group identities and non-group identities. Names must be unique across the same types.

*Parameters*

**identity : in TpPAMFQName**

Specifies the group Identity to be created.

**identityTypes : in TpStringList**

Specifies the group's associated types. Can be an empty array.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_IDENTITY\_EXISTS, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.2.5 Method deleteGroupIdentity()

Delete the specified group identity and all its related data. Upon successful completion, associated aliases and attribute instances are deleted from the system. The identity is also removed from all groups of which the identity is a member. The member identities of the group are not deleted.

*Parameters*

**identity : in TpPAMFQName**

Specifies the group Identity to be deleted.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.2.6 Method addToGroup()

Add an existing identity to a group identity. Both the group identity and the member identity to be added must have been created before this operation can be invoked. A member identity can be a group identity. Implementation must not allow cycles in memberships.

*Parameters*

**group : in TpPAMFQName**

Specifies the group Identity to which the member will be added.

**member : in TpPAMFQName**

Specifies the identity to be added as a member of the group.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_MEMBER\_EXISTS, P\_PAM\_UNKNOWN\_GROUP, P\_PAM\_UNKNOWN\_MEMBER, P\_PAM\_IS\_CYCLIC, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.2.7 Method removeFromGroup()

Remove an existing identity from the membership of a group identity.

#### *Parameters*

**group : in TpPAMFQName**

Specifies the Group Identity from which the member will be removed.

**identity : in TpPAMFQName**

Specifies the Identity to be removed as a member of the group.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_NOT\_MEMBER, P\_PAM\_UNKNOWN\_GROUP,  
P\_PAM\_UNKNOWN\_MEMBER, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.2.8 Method listMembers()

List the members of the specified group Identity.

Returns a list of members of the specified group Identity. An empty list is returned if the identity has no members.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the group Identity whose members are required.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

#### *Returns*

**TpPAMFQNameList**

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_GROUP, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.2.9 Method isGroupIdentity()

Check if the specified group identity exists. The method returns false for non-group identities.

Returns true if a group identity with the specified name exists, false otherwise.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the Identity to be checked.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpBoolean***Raises***TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL****8.1.2.10 Method listGroupMembership()**

List the Group Identities the specified Identity is a member of.

Returns a list of all groups the specified Identity is member of. An empty list is returned if the identity is not member of any group.

*Parameters***identity : in TpPAMFQName**

Specifies the Identity to be looked up.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpPAMFQNameList***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL****8.1.2.11 Method addAlias()**

Add an alias in the specified namespace to an existing Identity. The alias domain name must be specified in Alias parameter. The identity can be a group identity.

*Parameters***identity : in TpPAMFQName**

Specifies the Identity to which the alias will be added.

**alias : in TpPAMFQName**

Specifies the alias to be added.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises***TpCommonExceptions, P\_PAM\_ALIAS\_EXISTS, P\_PAM\_ALIAS\_NOT\_UNIQUE, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL****8.1.2.12 Method removeAliases()**

Remove the specified alias from an existing identity.

*Parameters***identity : in TpPAMFQName**

Specifies the Identity from which the alias will be deleted.

**alias : in TpPAMFQName**

Specifies the alias to be deleted.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_UNASSIGNED\_ALIAS, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.2.13 Method listAliases()

List the aliases of the specified Identity.

Returns a list containing all aliases to the specified Identity. A list with zero elements is returned if there are no aliases associated with the identity.

*Parameters*

**identity : in TpPAMFQName**

Specifies the Identity to be looked up.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns*

**TpPAMFQNameList**

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.2.14 Method lookupByAlias()

Find the identity with the specified alias in the specified alias domain.

Returns the identity that has the specified alias. Returns null if the alias is not assigned to any identity.

*Parameters*

**alias : in TpPAMFQName**

Specifies the alias to be looked up.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns*

**TpPAMFQName**

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_ALIAS, P\_PAM\_INVALID\_CREDENTIAL**



### 8.1.2.15 Method associateTypes()

Associate an identity instance with the specified types. The identity will be associated with instances of any attributes defined with each type. The initial values of the attributes will be as specified in the definition of the type attributes.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the name of the identity.

**identityTypes : in TpStringList**

Specifies the names of the type to be associated.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_TYPE\_ASSOCIATED, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.2.16 Method disassociateTypes()

Remove the association of a type with an identity instance. The definition of the type itself remains unaffected and the types may continue to be associated with other identities.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity.

**identityTypes : in TpStringList**

Specifies the names of the types to be removed.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_DISASSOCIATED\_TYPE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.2.17 Method listTypesOfIdentity()

List the types associated with the specified Identity.

Returns a list containing all types associated with the specified Identity. A list with zero elements is returned if there are no types associated with the identity.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the Identity to be looked up.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpPAMFQNameList***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL****8.1.2.18 Method hasType()**

Check if the specified identity has the named type associated with it.

Returns true if an identity with the specified name has the named type associated with it, false otherwise.

*Parameters***identity : in TpPAMFQName**

Specifies the Identity to be checked.

**typeName : in TpString**

Specifies the type to be checked for.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpBoolean***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL****8.1.2.19 Method getIdentityAttributes()**

Return the attributes associated with the identity. If the identity type is not specified, all associated types are assumed to be of interest.

The return value contains the list of specified attributes and their values. If the attributes parameter is an empty array, all attributes in the named identity are output.

*Parameters***identity : in TpPAMFQName**

Specifies the Identity whose attributes are to be accessed.

**identityType : in TpString**

Specifies the type of the identity with which the required attributes are associated. Is optional.

**attributeNames : in TpStringList**

List of attributes of interest. Can be an empty array.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpPAMAttributeList***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL****8.1.2.20 Method setIdentityAttributes()**

Modify the attributes associated with the named Identity. The input may contain a subset of the attributes of the named type. Only the specified attributes will be modified and the rest will remain unchanged. If the type is unspecified, any associated type will be assumed.

*Parameters***identity : in TpPAMFQName**

Specifies the Identity.

**identityType : in TpString**

Specifies the type of the identity for the operation. Is optional.

**attributes : in TpPAMAttributeList**

Contains the list of attributes and their values.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_UNKNOWN\_ATTRIBUTES, P\_PAM\_INVALID\_CREDENTIAL****8.1.3 Interface Class IpPAMAgentManagement**

Inherits from: IpInterface.

The purpose of this interface is to manage agent (that models a hardware or software device) names, communication capabilities and sets of attributes associated with agents. An implementation may map these methods to operations on existing directories or databases. Some implementations may choose to provide a read-only access to the agent information.

Data associated with an agent is captured in attributes associated with types. An implementation may map different type attributes to different underlying stores or directories.

The names of agents within a namespace must be unique.

This interface is meant for use by provisioning applications that establish and maintain agent names.

<<Interface>> IpPAMAgentManagement
<pre> createAgent (agentName : in TpPAMFQName, agentTypes : in TpStringList, authToken : in   TpPAMCredential) : void deleteAgent (agentName : in TpPAMFQName, authToken : in TpPAMCredential) : void isAgent (agentName : in TpPAMFQName, authToken : in TpPAMCredential) : TpBoolean enableCapabilities (agentName : in TpPAMFQName, capabilities : in TpPAMCapabilityList, authToken : in   TpPAMCredential) : void disableCapabilities (agentName : in TpPAMFQName, capabilities : in TpPAMCapabilityList, authToken : in   TpPAMCredential) : void listEnabledCapabilities (agentName : in TpPAMFQName, authToken : in TpPAMCredential) :   TpPAMCapabilityList listAllCapabilities (agentName : in TpPAMFQName, authToken : in TpPAMCredential) :   TpPAMCapabilityList isCapableOf (agentName : in TpPAMFQName, capability : in TpPAMCapability, authToken : in   TpPAMCredential) : TpBoolean associateTypes (agentName : in TpPAMFQName, agentTypes : in TpStringList, authToken : in   TpPAMCredential) : void disassociateTypes (agentName : in TpPAMFQName, agentTypes : in TpStringList, authToken : in   TpPAMCredential) : void listTypesOfAgent (agentName : in TpPAMFQName, authToken : in TpPAMCredential) : TpStringList hasType (agentName : in TpPAMFQName, typeName : in TpString, authToken : in TpPAMCredential) :   TpBoolean getAgentAttributes (agentName : in TpPAMFQName, agentType : in TpString, attributeNames : in   TpStringList, authToken : in TpPAMCredential) : TpPAMAttributeList setAgentAttributes (agentName : in TpPAMFQName, agentType : in TpString, attributes : in   TpPAMAttributeList, authToken : in TpPAMCredential) : void </pre>

### 8.1.3.1 Method createAgent()

Create an agent initialized with the specified capabilities.

#### *Parameters*

**agentName : in TpPAMFQName**

Specifies the name of Agent to be created.

**agentTypes : in TpStringList**

Specifies the types of the Agent to be created. Can be an empty list.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_AGENT\_EXISTS, P\_PAM\_UNKNOWN\_TYPE,  
P\_PAM\_INVALID\_CREDENTIAL**

**8.1.3.2 Method deleteAgent()**

Delete the specified Agent and all related data from the system.

*Parameters*

**agentName : in TpPAMFQName**

Specifies the name of Agent to be created.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_INVALID\_CREDENTIAL**

**8.1.3.3 Method isAgent()**

Check if the specified agent exists.

Returns true if an agent with the specified name exists, false otherwise.

*Parameters*

**agentName : in TpPAMFQName**

Specifies the Agent to be checked.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns*

**TpBoolean**

*Raises*

**TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL**

**8.1.3.4 Method enableCapabilities()**

Enable the specified capabilities of the agent.

*Parameters*

**agentName : in TpPAMFQName**

Specifies the Agent.

**capabilities : in TpPAMCapabilityList**

Specifies the capabilities to be enabled.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.3.5 Method disableCapabilities()

Remove the specified capabilities from the Agent.

*Parameters*

**agentName : in TpPAMFQName**

Specifies the Agent.

**capabilities : in TpPAMCapabilityList**

Specifies the communication mode to be disabled.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_NO\_CAPABILITY,  
P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.3.6 Method listEnabledCapabilities()

List the enabled capabilities for the specified Agent.

Returns the list of enabled capabilities for the Agent. Returns an empty list if no enabled capabilities exist for the agent.

*Parameters*

**agentName : in TpPAMFQName**

Specifies the Agent whose capabilities are to be listed.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns*

**TpPAMCapabilityList**

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.3.7 Method listAllCapabilities()

List the capabilities for the specified Agent.

Returns the list of capabilities for the Agent. Returns an empty list if no capabilities exist for the agent.

*Parameters*

**agentName : in TpPAMFQName**

Specifies the Agent whose capabilities are to be listed.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpPAMCapabilityList***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_INVALID\_CREDENTIAL****8.1.3.8 Method isCapableOf()**

Check if an agent has a particular capability that is currently enabled.

Returns true if the agent has the specified capability, false otherwise.

*Parameters***agentName : in TpPAMFQName**

Specifies the Agent to be checked.

**capability : in TpPAMCapability**

Capability to be checked.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpBoolean***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_INVALID\_CREDENTIAL****8.1.3.9 Method associateTypes()**

Associate an agent instance with the specified types.

*Parameters***agentName : in TpPAMFQName**

Specifies the name of Agent.

**agentTypes : in TpStringList**

Specifies the types of the Agent to be associated.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_UNKNOWN\_TYPE,  
P\_PAM\_TYPE\_ASSOCIATED, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.3.10 Method disassociateTypes()

Remove the association of a type with an agent instance. The definition of the type itself remains unaffected and the types may continue to be associated with other agents.

#### *Parameters*

**agentName** : in TpPAMFQName

Specifies the agent.

**agentTypes** : in TpStringList

Specifies the names of the types to be removed.

**authToken** : in TpPAMCredential

Of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions**, **P\_PAM\_UNKNOWN\_AGENT**, **P\_PAM\_DISASSOCIATED\_TYPE**,  
**P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.3.11 Method listTypesOfAgent()

List the types associated with the specified agent.

Returns the list containing all types associated with the specified agent. An empty list is returned if there are no types associated with the agent.

#### *Parameters*

**agentName** : in TpPAMFQName

Specifies the agent to be looked up.

**authToken** : in TpPAMCredential

Of the entity who wishes to do this operation.

#### *Returns*

**TpStringList**

#### *Raises*

**TpCommonExceptions**, **P\_PAM\_UNKNOWN\_AGENT**, **P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.3.12 Method hasType()

Check if the specified agent has the named type associated with it.

Returns true if an agent with the specified name has the named type associated with it.

#### *Parameters*

**agentName** : in TpPAMFQName

Specifies the Agent to be checked.

**typeName** : in TpString

Specifies the type to be checked for.



**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns*

**TpBoolean**

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.3.13 Method getAgentAttributes()

Return the attributes associated with the agent. If the type is not specified, all associated types are assumed.

Return value contains the list of specified attributes and their values. If the attributes parameter is an empty list, all attributes in the named agent are returned.

*Parameters*

**agentName : in TpPAMFQName**

Specifies the agent.

**agentType : in TpString**

Specifies the type of interest. Is optional.

**attributeNames : in TpStringList**

List of attributes of interest. Can be an empty list.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns*

**TpPAMAttributeList**

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.3.14 Method setAgentAttributes()

Modify the attributes associated with the named agent. The input may contain a subset of the attributes of the named type. Only the specified attributes will be modified and the rest will remain unchanged.

*Parameters*

**agentName : in TpPAMFQName**

Specifies the agent.

**agentType : in TpString**

Specifies the type of interest. Is optional.

**attributes : in TpPAMAttributeList**

Contains the list of attributes and their values.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_UNKNOWN\_TYPE,  
P\_PAM\_UNKNOWN\_ATTRIBUTES, P\_PAM\_INVALID\_CREDENTIAL**

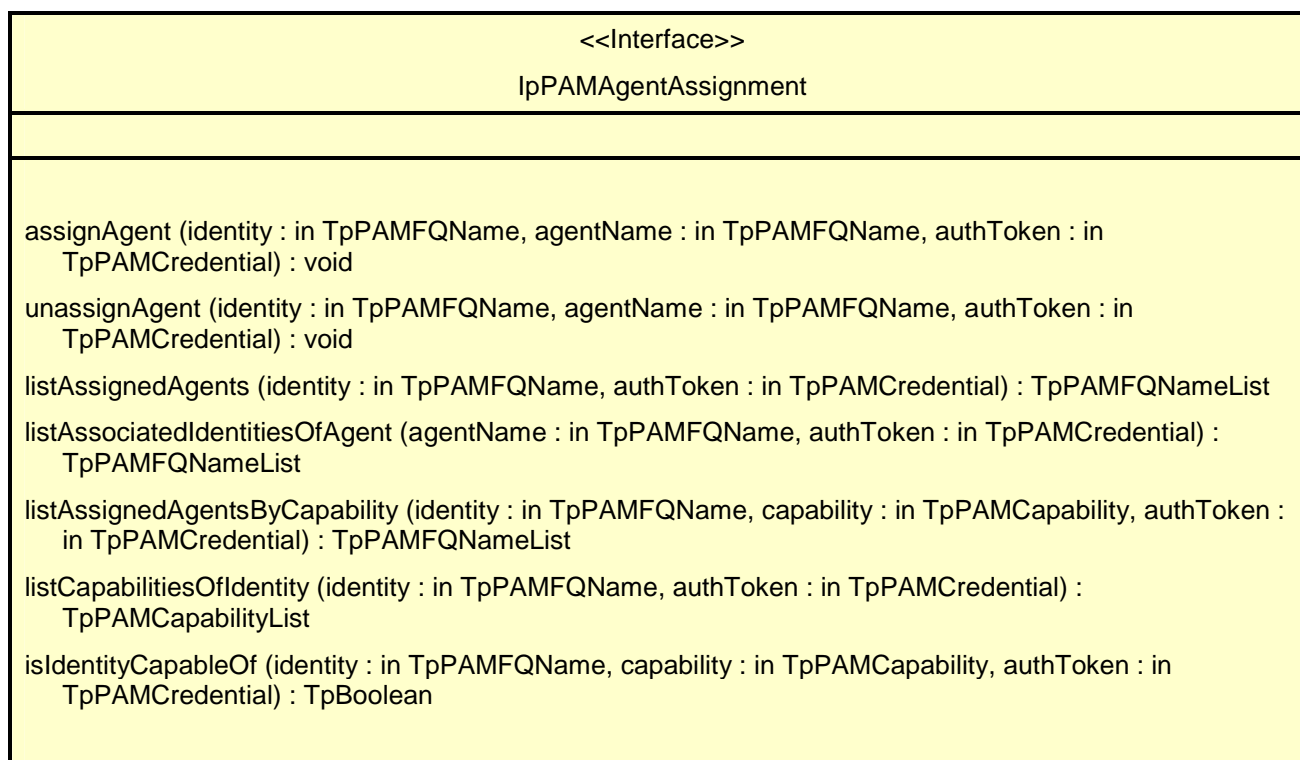
## 8.1.4 Interface Class IpPAMAgentAssignment

Inherits from: IpInterface.

The purpose of this interface is to manage the relationship between identities and the agents assigned to them. The identities inherit capabilities from the assignments of agents.

The implementation must maintain the integrity of the relationship between identities and agents across changes to both identities and agents. Implementations may map these methods to operations on existing directories and databases. Some implementations may provide a read-only access to this interface.

This interface is meant for use by provisioning applications that establish and maintain association of agents with identities.



### 8.1.4.1 Method assignAgent()

Assign an existing agent to an existing identity.

*Parameters*

**identity : in TpPAMFQName**

Specifies the identity to assign the agent to.

**agentName : in TpPAMFQName**

Specifies the Agent.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_INVALID\_CREDENTIAL**

#### 8.1.4.2 Method unassignAgent()

Unassign an agent from an existing identity. In effect, this deletes an existing relationship between an agent and an identity.

*Parameters*

**identity : in TpPAMFQName**

Specifies the identity to assign the agent to.

**agentName : in TpPAMFQName**

Specifies the Agent.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_UNKNOWN\_ASSIGNMENT, P\_PAM\_INVALID\_CREDENTIAL**

#### 8.1.4.3 Method listAssignedAgents()

List the Agents assigned to an identity.

Returns the list of agent names assigned to the identity. An empty list is returned if no agents are assigned to the identity.

*Parameters*

**identity : in TpPAMFQName**

Specifies the identity of interest.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns*

**TpPAMFQNameList**

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

#### 8.1.4.4 Method listAssociatedIdentitiesOfAgent()

List the identities that have the specified agent assigned to them.

Returns the list of identities that have been assigned the specified agent. Empty list is returned if no identities have been assigned this agent.

*Parameters***agentName : in TpPAMFQName**

Specifies the Agent.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpPAMFQNameList***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_INVALID\_CREDENTIAL****8.1.4.5 Method listAssignedAgentsByCapability()**

List the Agents assigned to an identity that match the specified capability.

Returns the list of agent names with the specified capability. An empty list is returned no agents are found.

*Parameters***identity : in TpPAMFQName**

Specifies the identity of interest.

**capability : in TpPAMCapability**

Specifies the capability of interest.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpPAMFQNameList***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL****8.1.4.6 Method listCapabilitiesOfIdentity()**

List the capabilities of an identity that it derives from its assigned Agents.

Returns the list of the identity's capabilities. Returns an empty list if no capabilities exist.

*Parameters***identity : in TpPAMFQName**

Specifies the identity of interest.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpPAMCapabilityList***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL****8.1.4.7 Method isIdentityCapableOf()**

Check if an identity has the specified capability derived from one or more agents assigned to it.

Returns true if the identity has this capability, false otherwise.

*Parameters***identity : in TpPAMFQName**

Specifies the identity of interest.

**capability : in TpPAMCapability**

Identifies the capability to check for.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpBoolean***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL****8.1.5 Interface Class IpPAMIdentityTypeManagement**

Inherits from: IpInterface.

This clause describes the programmatic interface to define the type schema for identities specifying the attributes associated with the type. These types can then be assigned to identities. PAM implementations may provide a set of pre-defined types. Identity type names and agent type names are in the same namespace and hence must be uniquely defined across both identities and agents. The attributes for identity types and agent types are in two different namespaces and hence an attribute name may be re-used with different characteristics for identities and agents.

<<Interface>> IpPAMIdentityTypeManagement
<pre> createlDentityAttribute (pAttribute : in TpPAMAttributeDef, authToken : in TpPAMCredential) : void deletelDentityAttribute (attributeName : in TpString, authToken : in TpPAMCredential) : void getlDentityAttributeDefinition (attributeName : in TpString, authToken : in TpPAMCredential) :   TpPAMAttributeDef listAlllDentityAttributes (authToken : in TpPAMCredential) : TpStringList createlDentityType (typeName : in TpString, attributeNames : in TpStringList, authToken : in   TpPAMCredential) : void deletelDentityType (typeName : in TpString, authToken : in TpPAMCredential) : void listlDentityTypes (authToken : in TpPAMCredential) : TpStringList addlDentityTypeAttributes (typeName : in TpString, attributeNames : in TpStringList, authToken : in   TpPAMCredential) : void removelDentityTypeAttributes (typeName : in TpString, attributeNames : in TpStringList, authToken : in   TpPAMCredential) : void listlDentityTypeAttributes (typeName : in TpString, authToken : in TpPAMCredential) : TpStringList           </pre>

### 8.1.5.1 Method createlDentityAttribute()

Create a definition of an identity attribute to specify its name and type.

#### *Parameters*

**pAttribute : in TpPAMAttributeDef**

Specifies the attribute to be created.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_ATTRIBUTE\_EXISTS, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.5.2 Method deletelDentityAttribute()

Delete the definition of an identity attribute.

#### *Parameters*

**attributeName : in TpString**

Specifies the attribute to be deleted.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

**8.1.5.3 Method getIdentityAttributeDefinition()**

Get the definition for the specified identity attribute.

Returns the definition of the specified attribute.

*Parameters*

**attributeName : in TpString**

Specifies the attribute.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Returns*

**TpPAMAttributeDef**

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

**8.1.5.4 Method listAllIdentityAttributes()**

List all known identity attributes.

Returns the list of attribute names defined so far. An empty array if no attributes have been defined.

*Parameters*

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Returns*

**TpStringList**

*Raises*

**TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL**

**8.1.5.5 Method createIdentityType()**

Specify a label as the name of an identity type.

*Parameters*

**typeName : in TpString**

Specifies the name of the type to be created.

**attributeNames : in TpStringList**

Specifies the list of attributes to be added.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_TYPE\_EXISTS, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

8.1.5.6 Method `deleteIdentityType()`

Delete a label as the name of an identity type. All identities that have this type are no longer associated with this type and consequently will no longer will have any attributes associated with this type.

*Parameters*

**typeName : in TpString**

Specifies the name of the type.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_INVALID\_CREDENTIAL**

8.1.5.7 Method `listIdentityTypes()`

List all known identity types.

Returns the list of known identity types. An empty array if no identity types have been defined.

*Parameters*

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Returns*

**TpStringList**

*Raises*

**TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL**

8.1.5.8 Method `addIdentityTypeAttributes()`

Add attribute definitions to the schema of an identity type that has already been defined.

*Parameters*

**typeName : in TpString**

Specifies the name of the type.

**attributeNames : in TpStringList**

List of attributes to be added to this type.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_ATTRIBUTE\_EXISTS, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**



### 8.1.5.9 Method `removeIdentityTypeAttributes()`

Delete attribute definitions from the schema of an identity type that has already been defined.

#### *Parameters*

**typeName** : in **TpString**

Specifies the name of the type.

**attributeNames** : in **TpStringList**

Specifies the list of attributes to be deleted.

**authToken** : in **TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions**, **P\_PAM\_UNKNOWN\_TYPE**, **P\_PAM\_UNKNOWN\_ATTRIBUTE**,  
**P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.5.10 Method `listIdentityTypeAttributes()`

List all attributes of an identity type.

Returns the list of attribute definitions for the named type. An empty array if no attributes have been defined for this type.

#### *Parameters*

**typeName** : in **TpString**

Specifies the name of the type.

**authToken** : in **TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Returns*

**TpStringList**

#### *Raises*

**TpCommonExceptions**, **P\_PAM\_UNKNOWN\_TYPE**, **P\_PAM\_INVALID\_CREDENTIAL**

## 8.1.6 Interface Class `IpPAMAgentTypeManagement`

Inherits from: `IpInterface`.

This clause describes the programmatic interface to define the type schema for agents specifying the attributes associated with the type. These types can then be assigned to agents. PAM implementations may provide a set of pre-defined types. Identity type names and agent type names are in the same namespace and hence must be uniquely defined across both identities and agents. The attributes for identity types and agent types are in two different namespaces and hence an attribute name may be re-used with different characteristics for identities and agents.

<<Interface>> IpPAMAgentTypeManagement
<pre> createAgentAttribute (pAttribute : in TpPAMAttributeDef, authToken : in TpPAMCredential) : void deleteAgentAttribute (attributeName : in TpString, authToken : in TpPAMCredential) : void getAgentAttributeDefinition (attributeName : in TpString, authToken : in TpPAMCredential) :   TpPAMAttributeDef listAllAgentAttributes (authToken : in TpPAMCredential) : TpStringList createAgentType (typeName : in TpString, attributeNames : in TpStringList, authToken : in   TpPAMCredential) : void deleteAgentType (typeName : in TpString, authToken : in TpPAMCredential) : void listAgentTypes (authToken : in TpPAMCredential) : TpStringList addAgentTypeAttributes (typeName : in TpString, attributeNames : in TpStringList, authToken : in   TpPAMCredential) : void removeAgentTypeAttributes (typeName : in TpString, attributeNames : in TpStringList, authToken : in   TpPAMCredential) : void listAgentTypeAttributes (typeName : in TpString, authToken : in TpPAMCredential) : TpStringList           </pre>

#### 8.1.6.1 Method createAgentAttribute()

Create a definition of an Agent attribute to specify its name and type.

##### *Parameters*

**pAttribute : in TpPAMAttributeDef**

Specifies the attribute to be created.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

##### *Raises*

**TpCommonExceptions, P\_PAM\_ATTRIBUTE\_EXISTS, P\_PAM\_INVALID\_CREDENTIAL**

#### 8.1.6.2 Method deleteAgentAttribute()

Delete the definition of an Agent attribute.

##### *Parameters*

**attributeName : in TpString**

Specifies the attribute to be deleted.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

**8.1.6.3 Method getAgentAttributeDefinition()**

Get the definition for the specified Agent attribute.

Returns the definition of the specified attribute.

*Parameters*

**attributeName : in TpString**

Specifies the attribute.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Returns*

**TpPAMAttributeDef**

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

**8.1.6.4 Method listAllAgentAttributes()**

List all known Agent attributes.

Returns the list of attribute names defined so far. An empty array if no attributes have been defined.

*Parameters*

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Returns*

**TpStringList**

*Raises*

**TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL**

**8.1.6.5 Method createAgentType()**

Specify a label as the name of an Agent type.

*Parameters*

**typeName : in TpString**

Specifies the name of the type to be created.

**attributeNames : in TpStringList**

Specifies the list of attributes to be added.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_TYPE\_EXISTS, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

**8.1.6.6 Method deleteAgentType()**

Delete a label as the name of an Agent type. All identities that have this type are no longer associated with this type and consequently will no longer will have any attributes associated with this type.

*Parameters*

**typeName : in TpString**

Specifies the name of the type.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_INVALID\_CREDENTIAL**

**8.1.6.7 Method listAgentTypes()**

List all known Agent types.

Returns the list of known Agent types. An empty array if no Agent types have been defined.

*Parameters*

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Returns*

**TpStringList**

*Raises*

**TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL**

**8.1.6.8 Method addAgentTypeAttributes()**

Add attribute definitions to the schema of an Agent type that has already been defined.

*Parameters*

**typeName : in TpString**

Specifies the name of the type.

**attributeNames : in TpStringList**

List of attributes to be added to this type.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_ATTRIBUTE\_EXISTS, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.6.9 Method removeAgentTypeAttributes()

Delete attribute definitions from the schema of an Agent type that has already been defined.

#### *Parameters*

**typeName** : in TpString

Specifies the name of the type.

**attributeNames** : in TpStringList

Specifies the list of attributes to be deleted.

**authToken** : in TpPAMCredential

Credential of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.6.10 Method listAgentTypeAttributes()

List all attributes of an Agent type.

Returns the list of attributes for the named type. An empty array if no attributes have been defined for this type.

#### *Parameters*

**typeName** : in TpString

Specifies the name of the type.

**authToken** : in TpPAMCredential

Credential of the entity who wishes to do this operation.

#### *Returns*

**TpStringList**

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_INVALID\_CREDENTIAL**

## 8.1.7 Interface Class IpPAMCapabilityManagement

Inherits from: IpInterface.

This clause describes the programmatic interface to define capability names.

Capabilities are names that define a property of an agent for which presence data may exist. Examples are voice, IM, SMS, WAP, etc. Agents can be assigned capabilities. Identities inherit capabilities from agents but cannot be directly assigned capabilities. Each capability is defined with an associated set of attributes. The attributes for each capability exist in their own namespace and hence an attribute name may be re-used with different characteristics across capabilities.

<<Interface>> IpPAMCapabilityManagement
<pre> createCapabilityAttribute (pAttribute : in TpPAMAttributeDef, authToken : in TpPAMCredential) : void deleteCapabilityAttribute (attributeName : in TpString, authToken : in TpPAMCredential) : void getCapabilityAttributeDefinition (attributeName : in TpString, authToken : in TpPAMCredential) :   TpPAMAttributeDef listAllCapabilityAttributes (authToken : in TpPAMCredential) : TpStringList createCapability (capabilityName : in TpPAMCapability, attributeNames : in TpStringList, authToken : in   TpPAMCredential) : void deleteCapability (capabilityName : in TpPAMCapability, authToken : in TpPAMCredential) : void listCapabilities (authToken : in TpPAMCredential) : TpPAMCapabilityList addCapabilityAttributes (capabilityName : in TpPAMCapability, attributeNames : in TpStringList, authToken :   in TpPAMCredential) : void removeCapabilityAttributes (capabilityName : in TpPAMCapability, attributeNames : in TpStringList,   authToken : in TpPAMCredential) : void listCapabilityAttributes (capabilityName : in TpPAMCapability, authToken : in TpPAMCredential) :   TpStringList assignCapabilitiesToType (agentType : in TpString, capabilities : in TpPAMCapabilityList, authToken : in   TpPAMCredential) : void unassignCapabilitiesFromType (agentType : in TpString, capabilities : in TpPAMCapabilityList, authToken :   in TpPAMCredential) : void listCapabilitiesOfType (agentType : in TpString, authToken : in TpPAMCredential) : TpPAMCapabilityList           </pre>

### 8.1.7.1 Method createCapabilityAttribute()

Create a definition of a Capability attribute to specify its name and type.

#### *Parameters*

**pAttribute : in TpPAMAttributeDef**

Specifies the attribute to be created.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_ATTRIBUTE\_EXISTS, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.7.2 Method deleteCapabilityAttribute()

Delete the definition of a Capability attribute.

#### *Parameters*

**attributeName : in TpString**

Specifies the attribute to be deleted.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.7.3 Method getCapabilityAttributeDefinition()

Get the definition for the specified Capability attribute.

Returns the definition of the specified attribute.

#### *Parameters*

**attributeName : in TpString**

Specifies the attribute.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Returns*

**TpPAMAttributeDef**

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.7.4 Method listAllCapabilityAttributes()

List all known Capability attributes.

Returns the list of attribute names defined so far. An empty array if no attributes have been defined.

#### *Parameters*

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Returns*

**TpStringList**

#### *Raises*

**TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.7.5 Method createCapability()

Specify a label as the name of a Capability type.

#### *Parameters*

**capabilityName : in TpPAMCapability**

Specifies the name of the type to be created.

**attributeNames : in TpStringList**

Specifies the list of attributes to be added.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_TYPE\_EXISTS, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.7.6 Method deleteCapability()

Delete a label as the name of a Capability. All agents that have this type are no longer associated with this type and consequently will no longer will have any attributes associated with this type.

#### *Parameters*

**capabilityName : in TpPAMCapability**

Specifies the name of the type.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.7.7 Method listCapabilities()

List all known Capability types.

Returns the list of known Capability types. An empty array if no Capability types have been defined.

#### *Parameters*

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Returns*

**TpPAMCapabilityList**

#### *Raises*

**TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL**



### 8.1.7.8 Method addCapabilityAttributes()

Add attribute definitions to the schema of a capability that has already been defined.

#### *Parameters*

**capabilityName** : in TpPAMCapability

Specifies the name of the type.

**attributeNames** : in TpStringList

List of attributes to be added to this type.

**authToken** : in TpPAMCredential

Credential of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_ATTRIBUTE\_EXISTS, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.7.9 Method removeCapabilityAttributes()

Delete attribute definitions from the schema of a Capability that has already been defined.

#### *Parameters*

**capabilityName** : in TpPAMCapability

Specifies the name of the type.

**attributeNames** : in TpStringList

Specifies the list of attributes to be deleted.

**authToken** : in TpPAMCredential

Credential of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.1.7.10 Method listCapabilityAttributes()

List all attributes of a capability.

Returns the list of attributes for the named Capability. An empty array if no attributes have been defined for this capability.

#### *Parameters*

**capabilityName** : in TpPAMCapability

Specifies the name of the type.

**authToken** : in TpPAMCredential

Credential of the entity who wishes to do this operation.

*Returns* **TpStringList** *Raises* **TpCommonExceptions, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_INVALID\_CREDENTIAL** **8.1.7.11 Method assignCapabilitiesToType()**

Assign capabilities to agent type.

*Parameters***agentType : in TpString**

Name of an agent type.

**capabilities : in TpPAMCapabilityList**

Specifies the list of capabilities to be associated.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises* **TpCommonExceptions, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_UNKNOWN\_CAPABILITY, P\_PAM\_INVALID\_CREDENTIAL** **8.1.7.12 Method unassignCapabilitiesFromType()**

Unassign capabilities from an agent type.

*Parameters***agentType : in TpString**

Name of an agent type.

**capabilities : in TpPAMCapabilityList**

Specifies the list of capabilities to be disassociated.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises* **TpCommonExceptions, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_UNKNOWN\_CAPABILITY, P\_PAM\_INVALID\_CREDENTIAL** **8.1.7.13 Method listCapabilitiesOfType()**

List capabilities assigned to an agent type.

Returns the list of capabilities assigned to the agent type.

*Parameters***agentType : in TpString**

Name of an agent type.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns*

**TpPAMCapabilityList**

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_INVALID\_CREDENTIAL**

## 8.2 PAM Access SCF Interface Classes

This service consists of the presence and availability query and update interfaces.

### 8.2.1 Interface Class IpPAMPresenceAvailabilityManager

Inherits from: IpService.

The purpose of this interface is to supply the various interfaces available in this service to the application and to provide the authentication credentials. This interface is the only discoverable interface from the framework.

All PAM methods optionally use an authentication token as a parameter since the outcome of the operations may depend on the entity requesting the operation. To enable this, the `getAuthToken()` method is used to obtain an implementation dependent token. An application that has authenticated itself with the OSA framework, can get an authentication token for itself. Alternatively, if the application is requesting PAM operations on behalf of multiple entities, authentication tokens may be requested for each such entity after providing any available data about the asker. These tokens can then be used repeatedly for operations within a session without further need to identify the asker.

<<Interface>> IpPAMPresenceAvailabilityManager
<pre> getAuthToken (askerData : in TpAttributeList) : TpPAMCredential obtainInterface (interfaceName : in TpPAMPresenceAvailabilityInterfaceName) : IpInterfaceRef &lt;&lt;new&gt;&gt; getAccessControl (identity : in TpPAMFQName, authToken : in TpPAMCredential) :   TpPAMAccessControlData &lt;&lt;new&gt;&gt; setAccessControl (identity : in TpPAMFQName, operation : in TpPAMPreferenceOp,   newAccessControl : in TpPAMAccessControlData, authToken : in TpPAMCredential) : void &lt;&lt;new&gt;&gt; activateService (identity : in TpPAMFQName, authToken : in TpPAMCredential) : void &lt;&lt;new&gt;&gt; deactivateService (identity : in TpPAMFQName, authToken : in TpPAMCredential) : void &lt;&lt;new&gt;&gt; isActiveIdentity (identity : in TpPAMFQName, authToken : in TpPAMCredential) : TpBoolean           </pre>

#### 8.2.1.1 Method getAuthToken()

Get an authentication token for access to the interface methods.

Returns an implementation-dependent authentication credential that can be verified.

*Parameters***askerData : in TpAttributeList**

Specifies information about the asker. Can be an empty array. The exact attributes in this list are dependent on the application. PAM reserves the attribute "name" with type TpPAMFQName to contain the identity of the asker if known.

*Returns***TpPAMCredential***Raises***TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL****8.2.1.2 Method obtainInterface()**

Obtain available interfaces from the service. The valid parameters for this method can be obtained from the service property P\_OBTAINABLE\_INTERFACES.

Returns the requested interface.

*Parameters***interfaceName : in TpPAMPresenceAvailabilityInterfaceName**

Specifies the name of the required interface.

*Returns***IpInterfaceRef***Raises***TpCommonExceptions, P\_PAM\_UNAVAILABLE\_INTERFACE****8.2.1.3 Method <<new>> getAccessControl()**

Get the access control associated with the data belonging to an identity. The data associated with an identity includes the static and dynamic attributes of an identity as well as data about agents associated with an identity.

This method should be used in conjunction with the setAccessControl method.

Returns the access control if previously specified for the identity. Is null if there is no access control associated.

*Parameters***identity : in TpPAMFQName**

Specifies the identity of interest.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpPAMAccessControlData***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

#### 8.2.1.4 Method <<new>> setAccessControl()

Set the access controls for the data associated with the specified identity. If the identity is Null, the access control is set for all identities (if authorized to do so). The data associated with an identity includes the static and dynamic attributes of an identity as well as data about agents associated with an identity.

Any existing access control will be modified based on the operation.

If the new access control is specified as Null for replace operation , an existing access control will be removed.

##### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity of interest.

**operation : in TpPAMPreferenceOp**

Specifies the operation to be performed with the specified preference.

**newAccessControl : in TpPAMAccessControlData**

Specifies the access controls to add.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

##### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

#### 8.2.1.5 Method <<new>> activateService()

Activate the service for the specified identity. If the identity is null, activate the service for ALL identities. The method succeeds if the service is already activated for the specified identity.

##### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity of interest. Can be NULL (empty string).

**authToken : in TpPAMCredential**

of the entity who wishes to do this operation.

##### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

#### 8.2.1.6 Method <<new>> deactivateService()

Deactivate the service for the specified identity. If the identity is null, deactivate the service for ALL identities. The method succeeds if the service is already deactivated for the specified identity.

##### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity of interest. Can be NULL (empty string).

**authToken : in TpPAMCredential**

of the entity who wishes to do this operation.

*Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL****8.2.1.7 Method <<new>> isActiveIdentity()**

Check if the service is active for the specified identity.

Returns true if the service is active for that identity, false otherwise.

*Parameters***identity : in TpPAMFQName**

Specifies the identity of interest. Cannot be NULL (empty string).

**authToken : in TpPAMCredential**

of the entity who wishes to do this operation.

*Returns***TpBoolean***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL****8.2.2 Interface Class IpPAMIdentityPresence**

Inherits from: IpInterface.

The purpose of this interface is to maintain the dynamic presence information of identity.

The underlying implementations may optimize the storage for this dynamic data rather than rely on a general-purpose directory or database when performance is an issue. Presence information for identities may be explicitly registered or may be implicitly derived from the underlying networks or presence information from agents associated with the identity.

This interface is meant for use by applications that register and/or maintain dynamic presence information associated with identities and accessible without the privacy or other controls established by availability preferences. These applications may not be aware of the name and the types of agents associated with the identity.

The presence information can be explicitly registered using the interface or the presence may come from information implicitly derived (e.g. using presence information of agents associated with the identity).

<<Interface>> IpPAMIdentityPresence
setIdentityPresence (identity : in TpPAMFQName, identityType : in TpString, attributes : in TpPAMAttributeList, authToken : in TpPAMCredential) : void setIdentityPresenceExpiration (identity : in TpPAMFQName, identityType : in TpString, attributeNames : in TpStringList, expiresIn : in TpPAMTimeInterval, authToken : in TpPAMCredential) : void getIdentityPresence (identity : in TpPAMFQName, identityType : in TpString, attributeNames : in TpStringList, authToken : in TpPAMCredential) : TpPAMAttributeList

### 8.2.2.1 Method setIdentityPresence()

Set identity's dynamic attributes.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity.

**identityType : in TpString**

Specifies the type of the identity.

**attributes : in TpPAMAttributeList**

Specifies the attributes to set.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.2.2.2 Method setIdentityPresenceExpiration()

Set or reset the expiration of an identity's named presence attributes. If the attributeNames parameter is an empty list, the expiration time of all attributes defined for the identity will have their expiration time changed.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity.

**identityType : in TpString**

Specifies the type of the identity.

**attributeNames : in TpStringList**

Specifies the names of the attributes. Can be an empty list.

**expiresIn : in TpPAMTimeInterval**

Specifies the number of seconds until the attributes expire. A value of -1 indicates no expiration.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.2.2.3 Method getIdentityPresence()

Retrieve presence attributes associated with an identity.

Return value contains the requested attributes of the named capability. If the attributes parameter is an empty array, all attributes of the named profile are included.

*Parameters***identity : in TpPAMFQName**

Specifies the identity.

**identityType : in TpString**

Specifies the type of the identity.

**attributeNames : in TpStringList**

Specifies the attributes of interest. Can be an empty list.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpPAMAttributeList***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

### 8.2.3 Interface Class IpPAMAvailability

Inherits from: IpInterface.

The purpose of the interface is to:

- Manage the preferences specified for the availability of an identity and, to
- Query for the availability of identities for specific capabilities.
- Query for attributes of interest from an identity.

Simple implementations may equate the availability of identities to presence of their agents with available status.

More complex implementations may consider, in addition, the preferences specified for availability as well as the attributes of the entity asking for availability.

The queries for availability are done for a specified context. A context is a set of attributes describing the situation for which availability is requested. PAM specifies one pre-defined context - Communication. The Communication context is used for availability for a specific mode of communication. Applications and PAM implementations may extend and provide additional contexts such as availability at a particular location, availability for a specific mode of communication at a given location, etc. The context information also includes any information about the asker as may be provided by the asker.

The specification defines two types of preference mechanisms although implementations may support additional mechanisms. The first mechanism consists of access control lists that specify identities that are allowed/denied to access information about the identity whose preference is being set. The second mechanism allow for an external application interface to be specified to check for access control as well as to compute availability.



<<Interface>> IpPAMAvailability
<pre> getAvailability (identity : in TpPAMFQName, pamContext : in TpPAMContext, attributeNames : in   TpStringList, authToken : in TpPAMCredential) : TpPAMAvailabilityProfileList getPreference (identity : in TpPAMFQName, pamContext : in TpPAMContext, authToken : in   TpPAMCredential) : TpPAMPreferenceData setPreference (identity : in TpPAMFQName, pamContext : in TpPAMContext, operation : in   TpPAMPreferenceOp, newPreference : in TpPAMPreferenceData, authToken : in TpPAMCredential) :   void </pre>

### 8.2.3.1 Method getAvailability()

Get the availability for an identity for a given context.

All contexts may optionally include an asker profile. Although PAM applications may decide what attributes to include in an asker profile, PAM implementations should not require such attributes to be present. The implementations should leave it to the availability computations to decide the availability based on the (partial) information provided.

It is also up to the availability computation to decide on the trustworthiness of the asker profile information based on the application, the credentials of the entity asking for availability and/or the credentials, if any, of the entity accessing the interface.

Returns a value containing a list of attributes as available to the asker in the requested context. If no information is available to the asker an empty list is returned.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity for which the availability is being requested.

**pamContext : in TpPAMContext**

Specifies the context for which the availability is requested.

**attributeNames : in TpStringList**

Specifies the attributes of interest. Can be an empty list to indicate all attributes.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

#### *Returns*

**TpPAMAvailabilityProfileList**

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

### 8.2.3.2 Method getPreference()

Get the availability preferences of an identity for the specified communication mode.

This method should be used in conjunction with the setPreference method.

Returns the preference for the named context if previously specified for the identity. Is null if there are no preferences associated.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity of interest.

**pamContext : in TpPAMContext**

Specifies the context for which the preferences are requested.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

#### *Returns*

**TpPAMPreferenceData**

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

### 8.2.3.3 Method setPreference()

Set the availability preferences for the specified identity for the specified context. If the identity is Null, the preference is set for all identities (if authorized to do so).

The existing preference will be modified based on the operation.

If the new preference is specified as Null for replace operation, any existing preferences for the specified context will be removed.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity with which the preference will be associated.

**pamContext : in TpPAMContext**

Specifies the capability to which this preference applies.

**operation : in TpPAMPreferenceOp**

Specifies the operation to be performed with the specified preference.

**newPreference : in TpPAMPreferenceData**

Specifies the availability preference to add.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions , P\_PAM\_UNKNOWN\_IDENTITY , P\_PAM\_INVALID\_CREDENTIAL**

## 8.2.4 Interface Class IpPAMAgentPresence

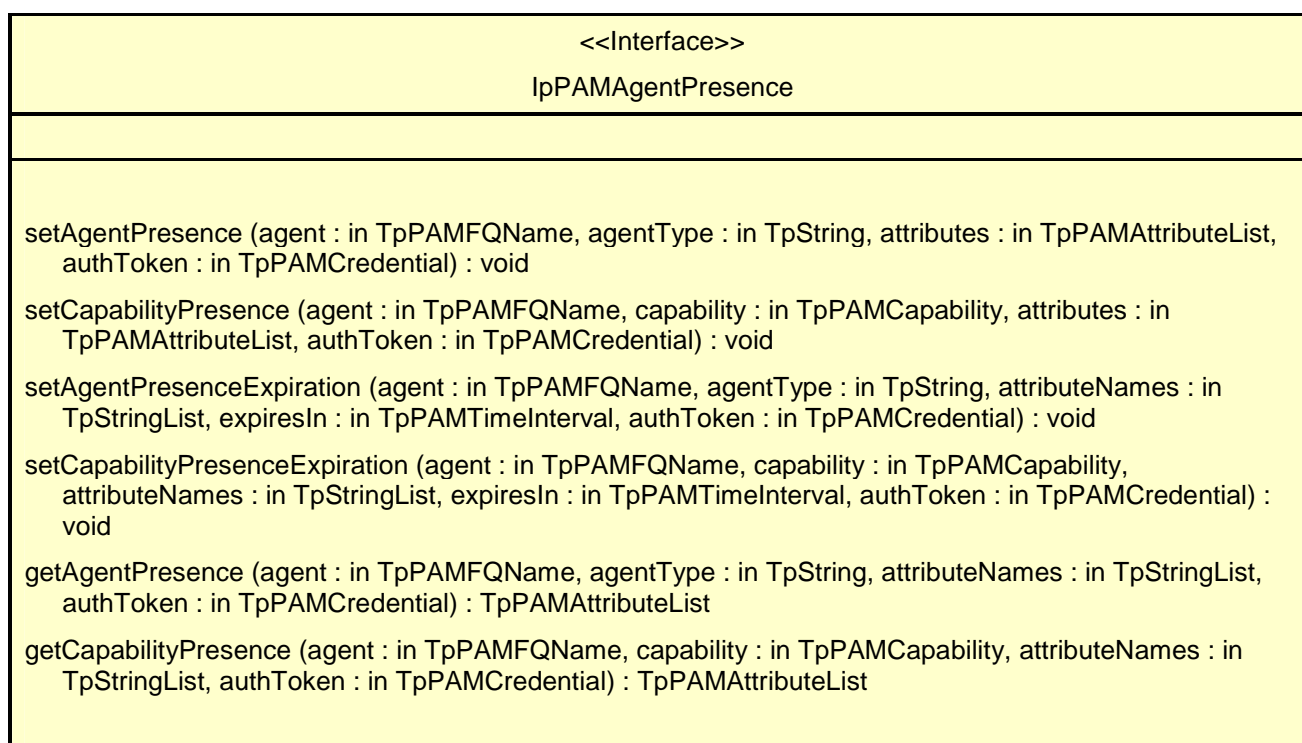
Inherits from: IpInterface.

The purpose of this interface is to maintain the dynamic presence information of agents.

The underlying implementations may optimize the storage for this dynamic data rather than rely on a general-purpose directory or database when performance is an issue. Agents may explicitly register the presence information or the presence information may be implicitly derived from the underlying networks.

The presence information is modelled through dynamic attributes. Sets of dynamic attributes can be defined per agent type (e.g. agent location, power status) or per agent capability (e.g. agent status for voice/messaging, communication address).

This interface is meant for use by applications that query and update agent presence information directly regardless of the identities to which the agent is assigned.



### 8.2.4.1 Method setAgentPresence()

Set presence attribute values for an agent.

*Parameters*

**agent : in TpPAMFQName**

Specifies the agent.

**agentType : in TpString**

Specifies the type of the agent.

**attributes : in TpPAMAttributeList**

Specifies the dynamic attributes to set.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_UNKNOWN\_TYPE,  
P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

#### 8.2.4.2 Method setCapabilityPresence()

Set presence attribute values for a set of capabilities of an agent.

*Parameters*

**agent : in TpPAMFQName**

Specifies the agent.

**capability : in TpPAMCapability**

Specifies which capability of the agent to set.

**attributes : in TpPAMAttributeList**

Specifies the dynamic attributes to set.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_UNKNOWN\_CAPABILITY,  
P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

#### 8.2.4.3 Method setAgentPresenceExpiration()

Set or reset the expiration of an agent's named presence profile.

*Parameters*

**agent : in TpPAMFQName**

Specifies the agent.

**agentType : in TpString**

Specifies the type of the agent.

**attributeNames : in TpStringList**

Specifies the names of the dynamic attributes. May be an empty array to indicate all dynamic attributes are to be affected.

**expiresIn : in TpPAMTimeInterval**

Specifies the number of seconds until the attributes expire. A value of -1 indicates no expiration. A value of 0 indicates immediate expiration.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_UNKNOWN\_TYPE,  
 P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

## 8.2.4.4 Method setCapabilityPresenceExpiration()

Set or reset the expiration of named presence attributes for a set of capabilities of an agent.

*Parameters*

**agent : in TpPAMFQName**

Specifies the agent.

**capability : in TpPAMCapability**

Specifies the type of the agent.

**attributeNames : in TpStringList**

Specifies the names of the dynamic attributes. May be an empty array to indicate all dynamic attributes are to be affected.

**expiresIn : in TpPAMTimeInterval**

Specifies the number of seconds until the attributes expire. A value of -1 indicates no expiration. A value of 0 indicates immediate expiration.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_NO\_CAPABILITY,  
 P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL**

## 8.2.4.5 Method getAgentPresence()

Retrieve named presence attributes for an agent.

Return value contains the requested dynamic attributes associated with the specified agent. If the attributeNames parameter is an empty list, all dynamic attributes of the specified agent are included.

*Parameters*

**agent : in TpPAMFQName**

Specifies the agent.

**agentType : in TpString**

Specifies the type of the agent.

**attributeNames : in TpStringList**

Specifies the dynamic attributes of interest. Can be an empty array to indicate all dynamic attributes are to be retrieved.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpPAMAttributeList***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_UNKNOWN\_TYPE, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL****8.2.4.6 Method getCapabilityPresence()**

Retrieve named presence attributes for a capability of an agent.

Return value contains the requested dynamic attributes associated with the specified agent. If the attributeNames parameter is an empty list, all dynamic attributes of the specified agent are included.

*Parameters***agent : in TpPAMFQName**

Specifies the agent.

**capability : in TpPAMCapability**

Specifies which capability of the agent for which attributes are desired.

**attributeNames : in TpStringList**

Specifies the dynamic attributes of interest. Can be an empty array to indicate all dynamic attributes are to be retrieved.

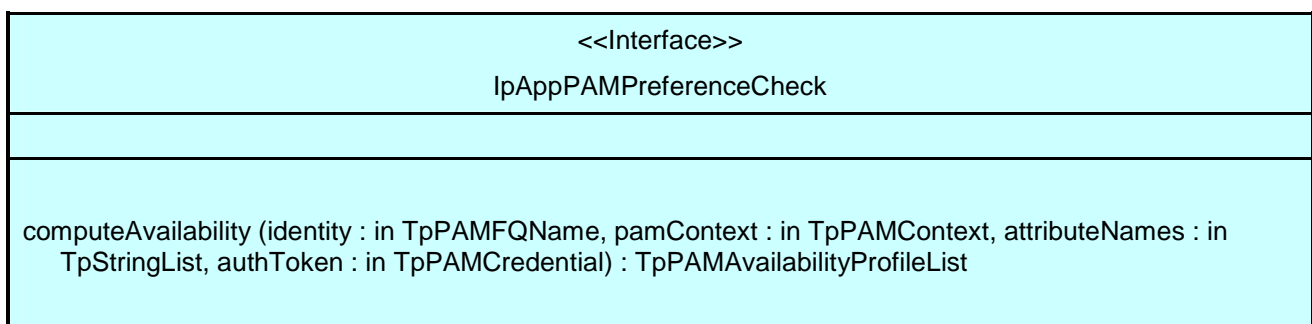
**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

*Returns***TpPAMAttributeList***Raises***TpCommonExceptions, P\_PAM\_UNKNOWN\_AGENT, P\_PAM\_UNKNOWN\_CAPABILITY, P\_PAM\_UNKNOWN\_ATTRIBUTE, P\_PAM\_INVALID\_CREDENTIAL****8.2.5 Interface Class IpAppPAMPreferenceCheck**

Inherits from: IpInterface.

The purpose of this interface is to provide methods to be called by the PAM service to check for access control or to compute availability using an implementation provided by an application. Instances of this interface are registered using the setPreference() method in the availability management interface.



### 8.2.5.1 Method computeAvailability()

Compute the availability for an identity for a given context. The data provided is the same as the data provided for the `getAvailability` call. The application implementing this interface uses the identity presence interface to get the current presence data and maintains its own user preferences to compute the availability.

Returns a value containing a list of attributes as available to the asker in the requested context. If no information is available to the asker an empty list is returned.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity for which the availability is being requested.

**pamContext : in TpPAMContext**

Specifies the context for which the availability is requested.

**attributeNames : in TpStringList**

Specifies the attributes of interest. Can be an empty list to indicate all attributes.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

#### *Returns*

**TpPAMAvailabilityProfileList**

## 8.3 PAM Event SCF Interface Classes

This service contains an interface for registering for notifications for events that occur within the PAM service.

### 8.3.1 Interface Class IpPAMEventManager

Inherits from: `IpService`.

The purpose of this interface is to supply the various interfaces available in this service to the application and to provide the authentication credentials. This interface is the only discoverable interface from the framework.

All PAM methods use an authentication token as a parameter since the outcome of the operations may depend on the entity requesting the operation. To enable this, the `getAuthToken()` method is used to obtain an implementation dependent token. An application that has authenticated itself with the OSA framework, can get an authentication token for itself. Alternatively, if the application is requesting PAM operations on behalf of multiple entities, authentication tokens may be requested for each such entity after providing any available data about the asker. These tokens can then be used repeatedly for operations within a session without further need to identify the asker.

<<Interface>> IpPAMEventManager
<pre> getAuthToken (askerData : in TpAttributeList) : TpPAMCredential obtainInterface (interfaceName : in TpPAMEventInterfaceName) : IpInterfaceRef &lt;&lt;new&gt;&gt; getAccessControl (identity : in TpPAMFQName, authToken : in TpPAMCredential) :   TpPAMAccessControlData &lt;&lt;new&gt;&gt; setAccessControl (identity : in TpPAMFQName, operation : in TpPAMPreferenceOp,   newAccessControl : in TpPAMAccessControlData, authToken : in TpPAMCredential) : void &lt;&lt;new&gt;&gt; activateService (identity : in TpPAMFQName, authToken : in TpPAMCredential) : void &lt;&lt;new&gt;&gt; deactivateService (identity : in TpPAMFQName, authToken : in TpPAMCredential) : void &lt;&lt;new&gt;&gt; isActiveIdentity (identity : in TpPAMFQName, authToken : in TpPAMCredential) : TpBoolean           </pre>

### 8.3.1.1 Method getAuthToken()

Get an authentication token for access to the interface methods.

Returns an implementation-dependent authentication credential that can be verified.

#### *Parameters*

**askerData : in TpAttributeList**

Specifies information about the asker. Can be an empty array. The exact attributes in this list are dependent on the application. PAM reserves the attribute "name" with type TpPAMFQName to contain the identity of the asker if known.

#### *Returns*

**TpPAMCredential**

#### *Raises*

**TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL**

### 8.3.1.2 Method obtainInterface()

Obtain available interfaces from the service. The valid parameters for this method can be obtained from the service property P\_OBTAINABLE\_INTERFACES.

Returns the requested interface.

#### *Parameters*

**interfaceName : in TpPAMEventInterfaceName**

Specifies the name of the required interface.

#### *Returns*

**IpInterfaceRef**

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNAVAILABLE\_INTERFACE**



### 8.3.1.3 Method <<new>> getAccessControl()

Get the access control associated with the data belonging to an identity. The data associated with an identity includes the static and dynamic attributes of an identity as well as data about agents associated with an identity.

This method should be used in conjunction with the setAccessControl method.

Returns the access control if previously specified for the identity. Is null if there is no access control associated.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity of interest.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

#### *Returns*

**TpPAMAccessControlData**

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

### 8.3.1.4 Method <<new>> setAccessControl()

Set the access controls for the data associated with the specified identity. If the identity is Null, the access control is set for all identities (if authorized to do so). The data associated with an identity includes the static and dynamic attributes of an identity as well as data about agents associated with an identity.

Any existing access control will be modified based on the operation.

If the new access control is specified as Null for replace operation , an existing access control will be removed.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity of interest.

**operation : in TpPAMPreferenceOp**

Specifies the operation to be performed with the specified preference.

**newAccessControl : in TpPAMAccessControlData**

Specifies the access controls to add.

**authToken : in TpPAMCredential**

Of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

### 8.3.1.5 Method <<new>> activateService()

Activate the service for the specified identity. If the identity is null, activate the service for ALL identities. The method succeeds if the service is already activated for the specified identity.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity of interest. Can be NULL (empty string).

**authToken : in TpPAMCredential**

of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

### 8.3.1.6 Method <<new>> deactivateService()

Deactivate the service for the specified identity. If the identity is null, deactivate the service for ALL identities. The method succeeds if the service is already deactivated for the specified identity.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity of interest. Can be NULL (empty string).

**authToken : in TpPAMCredential**

of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

### 8.3.1.7 Method <<new>> isActiveIdentity()

Check if the service is active for the specified identity. Returns true if the service is active for that identity, false otherwise.

#### *Parameters*

**identity : in TpPAMFQName**

Specifies the identity of interest. Cannot be NULL (empty string).

**authToken : in TpPAMCredential**

of the entity who wishes to do this operation.

#### *Returns*

**TpBoolean**

#### *Raises*

**TpCommonExceptions, P\_PAM\_UNKNOWN\_IDENTITY, P\_PAM\_INVALID\_CREDENTIAL**

## 8.3.2 Interface Class IpPAMEventHandler

Inherits from: IpInterface.

The purpose of this interface is to manage the registrations of interest in events and the registration of client interfaces for subsequent notification. All notifications in the present document are to be sent after the corresponding event has occurred and are asynchronous. An application must first register a notification interface with the service. It can then register interest in one or more events for this interface.

A failure or a reset of a PAM implementation may result in a loss of all prior event and interface registrations. The client application may need to confirm the continued registration of the notification interface and re-register if necessary.

For security and privacy purposes, a registration for an event is allowed if and only if the supplied credentials during registration is sufficient to have allowed access to the information related to the event through one or more of the PAM interface methods.

<<Interface>> IpPAMEventHandler
<pre> isRegistered (clientID : in TpPAMClientID, authToken : in TpPAMCredential) : TpBoolean registerAppInterface (appInterface : in IpAppPAMEventHandlerRef, authToken : in TpPAMCredential) :   TpPAMClientID registerForEvent (clientID : in TpPAMClientID, eventList : in TpPAMEventInfoList, validFor : in TpDuration,   authToken : in TpPAMCredential) : TpPAMEventID deregisterAppInterface (clientID : in TpPAMClientID, authToken : in TpPAMCredential) : void deregisterFromEvent (eventID : in TpPAMEventID, authToken : in TpPAMCredential) : void         </pre>

### 8.3.2.1 Method isRegistered()

Check if a client application interface is registered.

Returns True if the registration ID is still valid, False otherwise.

#### *Parameters*

**clientID : in TpPAMClientID**

Specifies the registration ID provided at registration.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Returns*

**TpBoolean**

#### *Raises*

**TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL**

### 8.3.2.2 Method registerAppInterface()

Register a client application's notification interface.

Returns an ID returned by the service that uniquely identifies this registration.

#### *Parameters*

**appInterface : in IpAppPAMEventHandlerRef**

Specifies the client notification interface.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Returns*

**TpPAMClientID**

#### *Raises*

**TpCommonExceptions, P\_PAM\_INVALID\_CREDENTIAL**

### 8.3.2.3 Method registerForEvent()

Register a client application's interest in one or more events.

Returns an ID returned by the service that uniquely identifies this registration for the event.

#### *Parameters*

**clientID : in TpPAMClientID**

Specifies the registration ID provided at registration.

**eventList : in TpPAMEventInfoList**

Specifies the events of interest.

**validFor : in TpDuration**

Specifies the interval in milliseconds until which the subscription is held and notifications provided. A time interval of 0 or negative values indicate a subscription that never expires until explicitly cancelled.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Returns*

**TpPAMEventID**

#### *Raises*

**TpCommonExceptions, P\_PAM\_NOT\_REGISTERED, P\_PAM\_INVALID\_CREDENTIAL**

### 8.3.2.4 Method deregisterAppInterface()

Unregister a client application's notification interface.

All registrations for events for this client registration are also removed.

#### *Parameters*

**clientID : in TpPAMClientID**

Specifies the registration ID provided at registration.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_NOT\_REGISTERED, P\_PAM\_INVALID\_CREDENTIAL**

### 8.3.2.5 Method deregisterFromEvent()

Unregister a client application's interest in an event.

#### *Parameters*

**eventID : in TpPAMEventID**

Specifies a prior event registration ID.

**authToken : in TpPAMCredential**

Credential of the entity who wishes to do this operation.

#### *Raises*

**TpCommonExceptions, P\_PAM\_NOT\_REGISTERED, P\_PAM\_INVALID\_CREDENTIAL**

## 8.3.3 Interface Class IpAppPAMEventHandler

Inherits from: IpInterface.

This is the interface that a client application must implement and register with the Event Service in order to be notified of events.

<<Interface>> IpAppPAMEventHandler
<pre> eventNotify (eventID : in TpPAMEventID, eventInfo : in TpPAMNotificationInfoList) : void eventNotifyErr (eventID : in TpPAMEventID, errorInfo : in TpPAMErrorInfo) : void </pre>

### 8.3.3.1 Method eventNotify()

Notify the occurrence of an event. The implementations will not attempt to re-notify on failure.

#### Parameters

**eventID : in TpPAMEventID**

Specifies a prior event registration ID.

**eventInfo : in TpPAMNotificationInfoList**

Contains the data about the events that occurred.

### 8.3.3.2 Method eventNotifyErr()

Notify an error in the event reporting. The error may concern all assignments, one whole assignment or a part of it. An eventNotify is sent after the error condition has passed away unless the event has been subsequently deregistered. Re-registration may only be needed in fatal system error cases. Note that in normal operation unavailable or protected pieces of presence information are delivered by the normal reporting methods instead of an error method.

#### Parameters

**eventID : in TpPAMEventID**

Specifies a prior event registration ID.

**errorInfo : in TpPAMErrorInfo**

Contains the data relating to the error.

## 9 State Transition Diagrams

There are no State Transition Diagrams for the Presence and Availability Management SCFs.

## 10 PAM Service Properties

The following table lists properties relevant to all the PAM SCFs.

Property	Type	Description
P_OBTAINABLE_INTERFACES	STRING_SET	The interfaces obtainable from the service
P_SUPPORTED_ATTRIBUTE_TAGS	STRING_SET	Lists the supported attribute tags defined by TpAttributeTagInfo
P_SUPPORTED_SIMPLE_ATTRIBUTE_TYPES	STRING_SET	Lists the supported attribute types defined by TpSimpleAttributeTypeInfo
P_SUPPORTED_STRUCTURED_ATTRIBUTE_TYPES	STRING_SET	Lists the supported attribute types defined by TpStructuredAttributeType, e.g. P_org/csapi/TpAddress
P_SUPPORTED_XML	STRING_SET	Lists the supported versions of XML specifications such as XML schema specifications (e.g. through URLs), XML versions (e.g. version 1.0) or XPath (e.g. version 1.0)

Implementations of the PAM APIs shall have the Service Properties set to the indicated values at a minimum:

```
P_SUPPORTED_ATTRIBUTE_TAGS = {
P_SIMPLE_TYPE
}
```

```
P_SUPPORTED_SIMPLE_ATTRIBUTE_TYPES = {
P_STRING,
P_FLOAT,
P_INT32
}
```

## 10.1 PAM Access Service

Implementations of the PAM Access APIs for 3GPP shall have the Service Properties set to the indicated values:

```
P_OBTAINABLE_INTERFACES = {
P_PAM_IDENTITY_PRESENCE,
P_PAM_AVAILABILITY
}
```

## 10.2 PAM Event Service

PAM Event service has the following property in addition to the above.

Property	Type	Description
P_EVENT_TYPES	INTEGER_SET	The pre-defined event types that can be registered for.

---

# 11 PAM Data Definitions

All data types referenced in the present document but not defined in this clause are common data definitions which may be found in ES 203 915-2.

## 11.1 Entity Address Definitions

### 11.1.1 TpPAMFQName

This is the same as TpURN and is used to name entities in PAM Access service.

### 11.1.2 TpPAMFQNameList

This is a `Numbered List of Data Elements` of type TpPAMFQName.

## 11.2 Attribute Data Definitions

### 11.2.1 TpPAMAttribute

This is a `Sequence of Data Elements` containing the attribute name, expiration time and value. This is derived from the common attribute type TpAttribute to add the expiration value for dynamic attributes.

Sequence Element Name	Sequence Element Type	Notes
AttributeName	TpString	The name of the attribute.
AttributeValue	TpAttributeValue	The typed value(s) for the attribute.
ExpiresIn	TpPAMTimeInterval	The interval in milliseconds in which the attribute values are valid. A time interval of PAM_MAX_LONGINT indicates static attribute values that never expire. A time interval of 0 or negative values indicate an expired value and the time for which it has expired.

## 11.2.2 TpPAMAttributeList

This is a `Numbered List of Data Elements` of type `TpPAMAttribute`.

## 11.2.3 TpPAMAttributeDef

This is a `Sequence of Data Elements` containing the definition of an attribute. This definition constitutes the "schema" for an attribute and contains fields to define the type and behaviour of a dynamic attribute. Each definition using these fields results in a `TpPAMAttribute` with the corresponding name and type and dynamic behaviour as defined by the remaining fields.

Sequence Element Name	Sequence Element Type	Notes
Name	TpString	Name of attribute.
Type	TpString	Type of attribute. Valid values for Type must include at least TpString, TpInt32 and TpFloat.
IsStatic	TpBoolean	True indicates that the attributes is always static and its values never expire. False indicates that the attribute can be dynamic and may contain values that expire.
IsRevertOnExpiration	TpBoolean	True indicates that the attribute reverts to the default value on expiration. False indicates that the attribute will not revert to the default value.
DefaultValues	TpAny	An attribute is always initialized with this value. If the <code>isRevertOnExpiration</code> attribute is set to true, a dynamic attribute that has expired while stored in a PAM implementation is reset to this value with the <code>expiresIn</code> interval set to <code>PAM_MAX_LONGINT</code> . The default attribute value is interpreted based on the value of the attribute Type.

## 11.2.4 TpPAMAttributeDefList

This is a `Numbered List of Data Elements` of type `TpPAMAttributeDef`.

## 11.3 Presence Data Definitions

### 11.3.1 TpPAMCapability

This defines the extensible communication capabilities. This data type is identical to a `TpString`, and is defined as a string of characters that specify the communication capabilities. The following strings are pre-defined.

Character String Value	Description
P_PAM_VOICE	Capability for voice calls
P_PAM_SMS	Capability for SMS
P_PAM_IM	Capability for Instant Messaging
P_PAM_MMS	Capability for Multi-media messaging

### 11.3.2 TpPAMCapabilityList

This is a `Numbered List of Data Elements` of type `TpPAMCapability`.



### 11.3.3 TpPAMPresenceData

This is a Sequence of Data Elements for a presence record. Since multiple presence data records can be associated with an identity, each distinct record is uniquely named.

Sequence Element Name	Sequence Element Type	Notes
Name	TpString	Name of presence data
PresenceAttributes	TpPAMAttributeList	Presence Attributes

### 11.3.4 TpPAMPresenceDataList

This is a Numbered List of Data Elements of type TpPAMPresenceData.

## 11.4 Pre-defined Presence type

### 11.4.1 Presentity

An identity type Presentity is pre-defined for all identities associated with the attribute PresenceProfile defined as:

Attribute Definition Field	Value
Name	PresenceProfile
Type	TpPAMPresenceData
IsStatic	False
IsRevertOnExpiration	False
DefaultValues	Null

## 11.5 Availability Data Definitions

### 11.5.1 TpPAMAvailabilityProfile

This is a Sequence of Data Elements containing the list of attribute values as determined by the definition of the context for which the availability is provided.

Sequence Element Name	Sequence Element Type	Notes
PrivacyCode	TpPAMPrivacyCode	Contains the privacy codes
AvailabilityData	TpPAMPresenceData	Contains a list of presence attributes

### 11.5.2 TpPAMAvailabilityProfileList

This is a Numbered List of Data Elements of type TpPAMAvailabilityProfile.

### 11.5.3 TpPAMPrivacyCode

This data type is identical to a TpString, and is defined as a string of characters that specify the privacy code for availability profiles. These codes are just indications of the privacy expected by the service and not are meant to be enforced by the service. Other Network operator specific codes may also be used, but should be preceded by the string "S\_". The following values are defined.

Character String Value	Description
PAM_CP_ASKER_ONLY	The profile is available to the asker only and should not be further transmitted
PAM_CP_AUTHORIZED	The profile can be provided by the asker to authorized entities
PAM_CP_UNLIMITED	The profile can be distributed without limits

## 11.6 Availability Context Data Definitions

Availability is always queried for in a specific context on behalf of an asker. There is one context for communication pre-defined in the present document.

### 11.6.1 TpPAMContext

This is a Sequence of Data Elements containing the data which defines the context in which an availability is queried and information about the asker that is requesting the data.

Sequence Element Name	Sequence Element Type	Notes
ContextData	TpPAMContextData	Contains the context name and the list of attributes that define the context. The attributes to be included for a given context are specified by the definition of the context.
AskerData	TpAttributeList	Contains information about the asker of availability. The exact attributes in this list are dependent on the application. PAM reserves the attribute "name" with type TpPAMFQName to contain the identity of the asker if known.

### 11.6.2 TpPAMContextName

This specifies the availability contexts.

Name	Value	Description
PAM_CONTEXT_ANY	0	Denotes any known context
PAM_CONTEXT_COMMUNICATION	1	Denotes a communication context

### 11.6.3 TpPAMContextData

This is a tagged choice of data elements that specifies the optional data that may be required to define a particular context.

Tag Element Type
TpPAMContextName

Tag Element Value	Choice Element Type	Choice Element Name
PAM_CONTEXT_ANY	None	Undefined
PAM_CONTEXT_COMMUNICATION	TpPAMCommunicationContext	CommunicationContext

### 11.6.4 TpPAMCommunicationContext

This is a Sequence of Data Elements containing the list of attribute values for defining a communication context.

Sequence Element Name	Sequence Element Type	Notes
CommunicationCapability	TpPAMCapability	Specifies the communication type for which the availability is requested

### 11.6.5 TpPAMContextList

This is a Numbered List of Data Elements of type TpPAMContext.

## 11.7 Credential data definitions

### 11.7.1 TpPAMCredential

This is the same as TpOctetSet. This data is opaque to the application and is implementation dependent. As this data is valid only in the context of a single session with the service and hence cannot be used across multiple services, there are no inter-operability issues here. The application simply uses the credential returned from the getAuthToken() method in all other methods that require the credentials.

## 11.8 Availability and Access Control Preference Data Definitions

PAM allows several types of preferences to be specified. It includes an access control list specifying who is allowed to check for presence or subscribe to presence data for each identity. It also includes an interface for an application to register an interface to do access control checks and availability computations outside of the presence service.

### 11.8.1 IpAppPAMPreferenceCheckRef

Defines a Reference to type IpAppPAMPreferenceCheck.

### 11.8.2 TpPAMAccessControlData

This is a Sequence of Data Elements for access control data.

Sequence Element Name	Sequence Element Type	Notes
DefaultPolicy	TpPAMACLDefault	Specifies whether the default policy is to allow or deny access for names not mentioned in the list
AllowList	TpPAMFQNameList	Specifies a list of identities to be allowed access
DenyList	TpPAMFQNameList	Specifies a list of identities to be denied access

### 11.8.3 TpPAMACLDefault

Defines the two possible default policies for access control.

Name	Value	Description
PAM_ACCESS_ALLOW	0	Allow access by default
PAM_ACCESS_DENY	1	Deny access by default

### 11.8.4 TpPAMPreferenceOp

This data type is identical to a TpString, and is defined as a string of characters that specify the operations to be performed with a preference. The following values are defined.

Character String Value	Description
PAM_ACCESS_ADD	Add the specified preference to the current preferences
PAM_ACCESS_DELETE	Delete the specified preference from the current preferences
PAM_ACCESS_REPLACE	Replace the current preferences with the specified preference

## 11.8.5 TpPAMPreferenceType

This specifies the names of privacy and access control mechanisms supported by the service.

Character String Value	Description
PAM_ACCESS_LIST	The control data contains additions or modifications to access control list of who is authorized to access the presence information or subscribe to it.
PAM_EXTERNAL_CONTROL	The access control and availability computations are done external to the presence service.

## 11.8.6 TpPAMPreferenceData

This is a tagged choice of data elements that specifies the preference data. The data depends on the type of preference being specified.

Tag Element Type
TpPAMPreferenceType

Tag Element Value	Choice Element Type	Choice Element Name
PAM_EXTERNAL_CONTROL	IpAppPAMPreferenceCheckRef	ExternalControllInterface

## 11.9 Time data definitions

### 11.9.1 TpPAMTimeInterval

This is identical to TpInt64.

## 11.10 Pre-defined Entity Types and Attributes

The present document pre-defines one identity type called "Presentity". The following constant can be used to refer to this Identity Type. All identities in the PAM service are associated with this identity type. For example, the identityType parameter in IpIdentityPresence and IpEventHandler methods take this as the value. This is also used in the event registration data structure (e.g. TpPAMAVCEventData) in the IdentityType field.

Character String Value	Description
P_PAM_PRESENTITY_TYPE	The pre-defined identity type called Presentity.

Every identity type in PAM can be defined with a set of attributes that are associated with all identities of that type. The following dynamic attributes are pre-defined as attributes of type TpPAMAttribute for the "Presentity" identity type and shall be supported as attributes of all identities in implementations of this service. These attributes are defined using TpPAMAttributeDef fields as follows.

AttributeName	AttributeType	IsStatic	IsRevertOnExpiration	DefaultValue	Description
P_SUBSCRIBER_STATUS	P_STRING	False	False	None	Specifies the status of the subscriber.
P_NETWORK_STATUS	P_STRING	False	False	None	Specifies the status of the network.
P_COMMUNICATION_MEANS	P_org/csapi/pam/TpPAMCapability	False	False	None	Specifies the means of communication. The type is TpPAMCapability.
P_CONTACT_ADDRESS	P_org/csapi/TpAddress	False	False	None	Address for communication
P_SUBSCRIBER_PROVIDED_LOCATION	P_STRING	False	False	None	Location information provided by subscriber. Is optional.
P_NETWORK_PROVIDED_LOCATION	P_STRING	False	False	None	Location information provided by subscriber. Is optional.
P_PRIORITY	P_INT32	False	False	None	Priority for communication.
P_OTHER_INFO	P_STRING	False	False	None	Additional information.

## 11.11 Interface name definitions

This clause defines the names to be used for obtaining interfaces from the corresponding service interfaces in each PAM SCF.

### 11.11.1 TpPAMProvisioningInterfaceName

This data type is identical to a TpString, and is defined as a string of characters that identify the names of the PAM Provisioning interfaces that are to be supported by the OSA API.

Character String Value	Description
P_PAM_IDENTITY_MANAGEMENT	The name for the PAM Identity Management interface
P_PAM_AGENT_MANAGEMENT	The name for the PAM Agent Management interface
P_PAM_AGENT_ASSIGNMENT	The name for the PAM Agent Assignment interface
P_PAM_IDENTITY_TYPE_MANAGEMENT	The name for the PAM Identity Type Management interface
P_PAM_AGENT_TYPE_MANAGEMENT	The name for the PAM Agent Type Management interface
P_PAM_CAPABILITY_TYPE_MANAGEMENT	The name for the PAM Capability Type Management interface

### 11.11.2 TpPAMPresenceAvailabilityInterfaceName

This data type is identical to a TpString, and is defined as a string of characters that identify the names of the PAM Access interfaces that are to be supported by the OSA API.

Character String Value	Description
P_PAM_IDENTITY_PRESENCE	The name for the PAM Identity Presence interface
P_PAM_AGENT_PRESENCE	The name for the PAM Agent Presence interface
P_PAM_AVAILABILITY	The name for the PAM Availability interface

### 11.11.3 TpPAMEventInterfaceName

This data type is identical to a TpString, and is defined as a string of characters that identify the names of the PAM Event management interfaces that are to be supported by the OSA API.

Character String Value	Description
P_PAM_EVENT_HANDLER	The name for the Event Handler interface.

## 11.12 Event data definitions

There are two sets of data structures used for events. One set is used by applications to provide information when registering for an event and the second set is used to supply information to the applications in the notifications when the events occur.

### 11.12.1 IpAppPAMEventHandlerRef

Defines a Reference to type IpAppPAMEventHandler.

### 11.12.2 TpPAMClientID

This is the same as TpInt32 and is used to identify, uniquely within an implementation, registration of an application interface for notification of events.

### 11.12.3 TpPAMEventID

This is the same as TpAssignmentID and is used to identify, uniquely within an implementation, a registration for a specific event.

### 11.12.4 TpPAMEventName

This data type identifies the values that specify the event names.

Name	Value	Description
PAM_CE_IDENTITY_PRESENCE_SET	0	Notify if the value of presence attributes of an identity is explicitly set
PAM_CE_AVAILABILITY_CHANGED	1	Notify if the availability of an identity changes
PAM_CE_WATCHERS_CHANGED	2	Notify if the current set of watchers change
PAM_CE_IDENTITY_CREATED	3	Notify if a new identity has been created
PAM_CE_IDENTITY_DELETED	4	Notify if an identity has been deleted
PAM_CE_GROUP_MEMBERSHIP_CHANGED	5	Notify if the membership of a group changes
PAM_CE_AGENT_CREATED	6	Notify if a new agent has been created
PAM_CE_AGENT_DELETED	7	Notify if an agent has been deleted
PAM_CE_AGENT_ASSIGNED	8	Notify if an agent is assigned to an identity
PAM_CE_AGENT_UNASSIGNED	9	Notify if an agent has been unassigned from an identity
PAM_CE_CAPABILITY_CHANGED	10	Notify if the capability of an identity changes
PAM_CE_AGENT_CAPABILITY_PRESENCE_SET	11	Notify if the value of presence attributes of an agent is explicitly set
PAM_CE_AGENT_PRESENCE_SET	12	Notify if the value of presence attributes of an agent is explicitly set

### 11.12.5 TpPAMEventNameList

This is a Numbered List of Data Elements of type TpPAMEventName.

Each event is defined by the data that applications must provide during registration using TpPAMEventInfo and data that is provided to the application during notification of such events using TpPAMNotificationInfo.

## 11.12.6 TpPAMEventInfo

This is a tagged choice of data elements that specifies the event data provided by applications while registering.

	Tag Element Type	
	TpPAMEventName	

Tag Element Value	Choice Element Type	Choice Element Name
PAM_CE_IDENTITY_PRESENCE_SET	TpPAMIPSEventData	IdentityPresenceSet
PAM_CE_AVAILABILITY_CHANGED	TpPAMAVCEventData	AvailabilityChanged
PAM_CE_WATCHERS_CHANGED	TpPAMWCEventData	WatchersChanged
PAM_CE_IDENTITY_CREATED	TpPAMICEventData	IdentityCreated
PAM_CE_IDENTITY_DELETED	TpPAMIDEventData	IdentityDeleted
PAM_CE_GROUP_MEMBERSHIP_CHANGED	TpPAMGMCEventData	GroupMembershipChanged
PAM_CE_AGENT_CREATED	TpPAMACEventData	AgentCreated
PAM_CE_AGENT_DELETED	TpPAMADEventData	AgentDeleted
PAM_CE_AGENT_ASSIGNED	TpPAMAAEventData	AgentAssigned
PAM_CE_AGENT_UNASSIGNED	TpPAMAUEventData	AgentUnassigned
PAM_CE_CAPABILITY_CHANGED	TpPAMCCEventData	CapabilityChanged
PAM_CE_AGENT_CAPABILITY_PRESENCE_SET	TpPAMACPSEventData	AgentCapabilityPresenceSet
PAM_CE_AGENT_PRESENCE_SET	TpPAMAPSEventData	AgentPresenceSet

## 11.12.7 TpPAMEventInfoList

This is a Numbered List of Data Elements of type TpPAMEventInfo.

## 11.12.8 TpPAMNotificationInfo

This is a tagged choice of data elements that specifies the notification data provided to the applications for each event.

	Tag Element Type	
	TpPAMEventName	

Tag Element Value	Choice Element Type	Choice Element Name
PAM_CE_IDENTITY_PRESENCE_SET	TpPAMIPSNNotificationData	IdentityPresenceSetNotify
PAM_CE_AVAILABILITY_CHANGED	TpPAMAVCNNotificationData	AvailabilityChangedNotify
PAM_CE_WATCHERS_CHANGED	TpPAMWCNNotificationData	WatchersChangedNotify
PAM_CE_IDENTITY_CREATED	TpPAMICNNotificationData	IdentityCreatedNotify
PAM_CE_IDENTITY_DELETED	TpPAMIDNNotificationData	IdentityDeletedNotify
PAM_CE_GROUP_MEMBERSHIP_CHANGED	TpPAMGMCNNotificationData	GroupMembershipChangedNotify
PAM_CE_AGENT_CREATED	TpPAMACNNotificationData	AgentCreatedNotify
PAM_CE_AGENT_DELETED	TpPAMADNNotificationData	AgentDeletedNotify
PAM_CE_AGENT_ASSIGNED	TpPAMAANNotificationData	AgentAssignedNotify
PAM_CE_AGENT_UNASSIGNED	TpPAMAUNNotificationData	AgentUnassignedNotify
PAM_CE_CAPABILITY_CHANGED	TpPAMCCNNotificationData	CapabilityChangedNotify
PAM_CE_AGENT_CAPABILITY_PRESENCE_SET	TpPAMACPSNotificationData	AgentCapabilityPresenceSetNotify
PAM_CE_AGENT_PRESENCE_SET	TpPAMAPSNNotificationData	AgentPresenceSetNotify

## 11.12.9 TpPAMNotificationInfoList

This is a Numbered List of Data Elements of type TpPAMNotificationInfo.

## 11.12.10 PAM\_CE\_IDENTITY\_CREATED

Notify if a new identity has been created. Notifications for creation of multiple identities are bunched into a single notification whenever possible. A notification of this event is NOT sent for new association of types with an existing identity.

### 11.12.10.1 TpPAMICEventData

This is a Sequence of Data Elements to specify the input data for subscribing to identity creations. The event is registered for changes in any agents of the named type. If no identity types are named, then the event is registered for all identity types.

Sequence Element Name	Sequence Element Type	Notes
IdentityType	TpStringList	Specifies the type of the identities for which this notification is requested. Can be an empty array if notification required for identities of any type.

### 11.12.10.2 TpPAMICNotificationData

This is a Sequence of Data Elements to specify the data that is provided in the notifications for identity creation events.

Sequence Element Name	Sequence Element Type	Notes
Identities	TpPAMFQNameList	Contains the names of the identities that have been created.

## 11.12.11 PAM\_CE\_IDENTITY\_DELETED

Notify if an identity has been deleted. Notifications for deletion of multiple identities are bunched into a single notification whenever possible. A notification of this event is NOT sent for removing association of types with an existing identity.

### 11.12.11.1 TpPAMIDEventData

This is a Sequence of Data Elements to specify the input data for subscribing to identity deletions. The event is registered for changes in any of the named identities. If no identities are named, then the event is registered for all agents.

Sequence Element Name	Sequence Element Type	Notes
IdentityName	TpPAMFQNameList	Specifies the name of the identity whose deletion is to be notified. Can be an empty array.
IdentityType	TpStringList	Specifies the type of the identity for which this notification is requested if identityName is an empty array. Can be an empty array if notification required for identities of any type.

### 11.12.11.2 TpPAMIDNotificationData

This is a Sequence of Data Elements to specify the data that is provided in the notifications for identity deletion events.

Sequence Element Name	Sequence Element Type	Notes
Identities	TpPAMFQNameList	Contains the names of the identities that have been deleted.



## 11.12.12 PAM\_CE\_GROUP\_MEMBERSHIP\_CHANGED

Notify if the membership of a group changes. Notifications for changes to multiple groups are bunched into a single notification whenever possible.

### 11.12.12.1 TpPAMGMCEventData

This is a Sequence of Data Elements to specify the input data for subscribing to group membership changes. The event is registered for changes in any of the named groups. If no groups are named, then the event is registered for all groups.

Sequence Element Name	Sequence Element Type	Notes
GroupName	TpPAMFQNameList	Specifies the name of the group for which the change is to be notified. Can be an empty array if notifications are required for any group.
GroupType	TpStringList	Specifies the type of the group for which this notification is requested if the groupName is specified as an empty array. Can be an empty array if notification required for groups of any type.

### 11.12.12.2 TpPAMGMCNotificationData

This is a Sequence of Data Elements to specify the data that is provided in the notifications for group membership changes.

Sequence Element Name	Sequence Element Type	Notes
Groups	TpPAMFQNameList	Contains the names of the groups that have been changed.

## 11.12.13 PAM\_CE\_AGENT\_CREATED

Notify if a new agent has been created. Notifications for creation of multiple agents are bunched into a single notification whenever possible. The notification for this event is NOT sent for new associations of types with agents.

### 11.12.13.1 TpPAMACEventData

This is a Sequence of Data Elements to specify the input data for subscribing to agent creations. The event is registered for changes in any agents of the named type. If no agent types are named, then the event is registered for all agent types.

Sequence Element Name	Sequence Element Type	Notes
AgentType	TpStringList	Specifies the type of the agents for which this notification is requested. Can be an empty array if notification required for agents of any type.

### 11.12.13.2 TpPAMACNotificationData

This is a Sequence of Data Elements to specify the data that is provided in the notifications for agent creation events.

Sequence Element Name	Sequence Element Type	Notes
Agents	TpPAMFQNameList	Contains the names of the agents that have been created.

## 11.12.14 PAM\_CE\_AGENT\_DELETED

Notify if an agent has been deleted. Notifications for deletion of multiple agents are bunched into a single notification whenever possible. This event notification is NOT sent for disassociating a type from an agent.

### 11.12.14.1 TpPAMADEventData

This is a Sequence of Data Elements to specify the input data for subscribing to agent deletions. The event is registered for changes in any of the named agents. If no agents are named, then the event is registered for all agents.

Sequence Element Name	Sequence Element Type	Notes
AgentName	TpPAMFQNameList	Specifies the name of the agent whose deletion is to be notified. Can be an empty array.
AgentType	TpStringList	Specifies the type of the agent for which this notification is requested if agentName is an empty array. Can be an empty array if notification required for agents of any type.

### 11.12.14.2 TpPAMADNotificationData

This is a Sequence of Data Elements to specify the data that is provided in the notifications for agent deletion events.

Sequence Element Name	Sequence Element Type	Notes
Agents	TpPAMFQNameList	Contains the names of the agents that have been deleted.

## 11.12.15 PAM\_CE\_AGENT\_ASSIGNED

Notify if an agent is assigned to an identity.

### 11.12.15.1 TpPAMAAEventData

This is a Sequence of Data Elements to specify the input data for subscribing to agent assignments from an identity. The event is registered for changes in any of the named agents. If no agents are named, then the event is registered for any agent.

Sequence Element Name	Sequence Element Type	Notes
IdentityName	TpPAMFQNameList	Specifies the name of the identity for which the assignment is to be notified. Can be an empty array if notification is required for any identity instance.
IdentityType	TpStringList	Specifies the type of the identity for which this notification is requested if the identityName is specified as an empty array. Can be an empty array if notification required for identities of any type.
AgentName	TpPAMFQNameList	Specifies the name of the agent whose assignment is to be notified. Can be an empty array.
AgentType	TpStringList	Specifies the type of the agent for which this notification is requested if agentName is an empty array. Can be an empty array if notification required for agents of any type.

### 11.12.15.2 TpPAMAANotificationData

This is a Sequence of Data Elements to specify the data that is provided in the notifications for agent assignment events.

Sequence Element Name	Sequence Element Type	Notes
Identity	TpPAMFQName	Contains the name of the identity to whom an agent has been assigned.
Agent	TpPAMFQName	Contains the name of the agent that has been assigned.

### 11.12.16 PAM\_CE\_AGENT\_UNASSIGNED

Notify if an agent has been unassigned from an identity.

#### 11.12.16.1 TpPAMAUEventData

This is a Sequence of Data Elements to specify the input data for subscribing to agent unassignments from an identity. The event is registered for changes in any of the named agents. If no agents are named, then the event is registered for all assigned agents.

Sequence Element Name	Sequence Element Type	Notes
IdentityName	TpPAMFQNameList	Specifies the name of the identity for which the unassignment is to be notified. Can be an empty array if notification is required for any identity instance.
IdentityType	TpStringList	Specifies the type of the identity for which this notification is requested if the identityName is specified as an empty array. Can be an empty array if notification required for identities of any type.
AgentName	TpPAMFQNameList	Specifies the name of the agent whose unassignment is to be notified. Can be an empty array.
AgentType	TpStringList	Specifies the type of the agent for which this notification is requested if agentName is an empty array. Can be an empty array if notification required for agents of any type.

#### 11.12.16.2 TpPAMAUNotificationData

This is a Sequence of Data Elements to specify the data that is provided in the notifications for agent unassignment events.

Sequence Element Name	Sequence Element Type	Notes
Identity	TpPAMFQName	Contains the name of the identity from whom an agent has been unassigned.
Agent	TpPAMFQName	Contains the name of the agent that has been unassigned.

### 11.12.17 PAM\_CE\_CAPABILITY\_CHANGED

Notify if the capability of an identity changes.

### 11.12.17.1 TpPAMCCEventData

This is a Sequence of Data Elements to specify the input data for subscribing to identity capability changed events. The event is registered for changes in any of the named capabilities. If no capabilities are named, then the event is registered for all capabilities.

Sequence Element Name	Sequence Element Type	Notes
IdentityName	TpPAMFQNameList	Specifies the name of the identity for which the capability change is to be notified. Can be an empty array if notification is required for any identity instance.
IdentityType	TpStringList	Specifies the type of the identity for which this notification is requested if the identityName is specified as an empty array. Can be an empty array if notification required for identities of any type.
Capabilities	TpPAMCapabilityList	Specifies the capabilities of interest. Can be an empty array if notifications are required for any capability.

### 11.12.17.2 TpPAMCCNotificationData

This is a Sequence of Data Elements to specify the data that is provided in the notifications for capability change events.

Sequence Element Name	Sequence Element Type	Notes
Identity	TpPAMFQName	Contains the name of the identity whose capability has changed.
Capabilities	TpPAMCapabilityList	Contains the capabilities that have changed (i.e. added or removed).

## 11.12.18 PAM\_CE\_AGENT\_CAPABILITY\_PRESENCE\_SET

Notify if the value of capability presence attributes of an agent is set. Expiration of the dynamic attributes does not trigger this notification.

### 11.12.18.1 TpPAMACPSEventData

This is a Sequence of Data Elements to specify the input data for subscribing to agent capability presence set events. The event is registered for changes in any of the named attributes. If no attributes are named, then the event is registered for all attributes in the presence information.

Sequence Element Name	Sequence Element Type	Notes
AgentName	TpPAMFQNameList	Specifies the name of the agent for which the capability presence change is to be notified. Can be an empty array if notification is required for any agent instance.
AgentType	TpStringList	Specifies the type of the agent for which this notification is requested if the agentName is specified as an empty array. Can be an empty array if notification required for agents of any type.
Capabilities	TpPAMCapabilityList	Specifies the capabilities of interest. Can be an empty array if notifications are required for any capability.
AttributeNames	TpStringList	Specifies attributes of interest. Can be an empty array
ReportingPeriod	TpPAMTimeInterval	Specifies the interval for periodic reporting (regardless of change). If -1, the event notification happens only on a change. If 0, there is a single immediate notification.

### 11.12.18.2 TpPAMACPSNotificationData

This is a Sequence of Data Elements to specify the data that is provided in the notifications for capability presence set events.

Sequence Element Name	Sequence Element Type	Notes
Agent	TpPAMFQName	Contains the name of the agent whose capability presence has changed.
Capability	TpPAMCapability	Specifies the capability for which the presence has changed.
AttributeNames	TpStringList	Contains the attribute names that have changed in value.

### 11.12.19 PAM\_CE\_AGENT\_PRESENCE\_SET

Notify if the value of presence attributes of an agent is set. Expiration of the dynamic attributes does not trigger this notification.

#### 11.12.19.1 TpPAMAPSEventData

This is a Sequence of Data Elements to specify the input data for subscribing to agent presence set events. The event is registered for changes in any of the named attributes. If no attributes are named, then the event is registered for all attributes in the presence information.

Sequence Element Name	Sequence Element Type	Notes
AgentName	TpPAMFQNameList	Specifies the name of the agent for which the assignment is to be notified. Can be an empty array if notification is required for any agent instance.
AgentType	TpStringList	specifies the type of the agent for which this notification is requested if the agentName is specified as an empty array. Can be an empty array if notification required for agents of any type.
AttributeNames	TpStringList	Specifies attributes of interest. Can be an empty array.
ReportingPeriod	TpPAMTimeInterval	Specifies the interval for periodic reporting (regardless of change). If -1, the event notification happens only on a change. If 0, there is a single immediate notification.

#### 11.12.19.2 TpPAMAPSNotificationData

This is a Sequence of Data Elements to specify the data that is provided in the notifications for agent presence set events.

Sequence Element Name	Sequence Element Type	Notes
Agent	TpPAMFQName	Contains the name of the agent whose capability has changed
AttributeNames	TpStringList	Contains the attribute names that have changed in value

### 11.12.20 PAM\_CE\_IDENTITY\_PRESENCE\_SET

Notify if the value of presence attributes of an identity is set. Expiration of the dynamic attributes does not trigger this notification.

### 11.12.20.1 TpPAMIPSEventData

This is a Sequence of Data Elements to specify the input data for subscribing to identity presence set events. The event is registered for changes in any of the named attributes. If no attributes are named, then the event is registered for all attributes in the presence information.

Sequence Element Name	Sequence Element Type	Notes
IdentityName	TpPAMFQNameList	Specifies the name of the identity for which the assignment is to be notified. Can be an empty array if notification is required for any identity instance.
IdentityType	TpStringList	Specifies the type of the identity for which this notification is requested if the identityName is specified as an empty array. Can be an empty array if notification required for identities of any type.
AttributeNames	TpStringList	Specifies attributes of interest. Can be an empty array.
ReportingPeriod	TpPAMTimeInterval	Specifies the interval for periodic reporting (regardless of change). If -1, the event notification happens only on a change. If 0, there is a single immediate notification even if there is no change. For all other values, there is a periodic notification at the specified time interval regardless of change.

### 11.12.20.2 TpPAMIPSNotificationData

This is a Sequence of Data Elements to specify the data that is provided in the notifications for identity presence set events.

Sequence Element Name	Sequence Element Type	Notes
Identity	TpPAMFQName	Contains the name of the identity whose capability has changed.
Attributes	TpPAMPresenceDataList	Contains the attributes that have changed in value.

### 11.12.21 PAM\_CE\_AVAILABILITY\_CHANGED

Notify if the availability of an identity changes. The event is registered for changes in any of the named attributes. If no attributes are named, then the event is registered for all attributes in the presence information.

### 11.12.21.1 TpPAMAVCEventData

This is a Sequence of Data Elements to specify the input data for subscribing to availability changed events.

Sequence Element Name	Sequence Element Type	Notes
IdentityName	TpPAMFQNameList	Specifies the name of the identity for which the assignment is to be notified. Can be an empty array if notification is required for any identity instance.
IdentityType	TpStringList	Specifies the type of the identity for which this notification is requested if the identityName is specified as an empty array. Can be an empty array if notification required for identities of any type.
PAMContext	TpPAMContextList	Specifies the context in which the availability is to be monitored. Cannot be an empty array.
AttributeNames	TpStringList	Specifies attributes of interest. Can be an empty array.
ReportingPeriod	TpPAMTimeInterval	Specifies the interval for periodic reporting (regardless of change). If -1, the event notification happens only on a change. If 0, there is a single immediate notification even if there is no change. For all other values, there is a periodic notification at the specified time interval regardless of change.

### 11.12.21.2 TpPAMAVCNotificationData

This is a Sequence of Data Elements to specify the data that is provided in the notifications for availability changed events.

Sequence Element Name	Sequence Element Type	Notes
Identity	TpPAMFQName	Contains the name of the identity whose capability has changed.
Availability	TpPAMAvailabilityProfileList	Contains the availability information that has changed.

## 11.12.22 PAM\_CE\_WATCHERS\_CHANGED

Notify if list of watchers for any event changed.

### 11.12.22.1 TpPAMWCEventData

This is a Sequence of Data Elements to specify the input data for subscribing to watchers changed events.

Sequence Element Name	Sequence Element Type	Notes
Events	TpPAMEventNameList	Specifies the list of events for which the watchers are to be monitored. Can be an empty array if notification is required for watchers to any event.
IdentityName	TpPAMFQNameList	Specifies the name of an identity for whom the change in watchers is to be notified. Can be an empty array if notification is required for any identity instance.
IdentityType	TpStringList	Specifies the type of the identity for which this notification is requested if the identityName is specified as an empty array. Can be an empty array if notification required for identities of any type.
ReportingPeriod	TpPAMTimeInterval	Specifies the interval for periodic reporting (regardless of change). If -1, the event notification happens only on a change. If 0, there is a single immediate notification even if there is no change. For all other values, there is a periodic notification at the specified time interval regardless of change.

### 11.12.22.2 TpPAMWCNotificationData

This is a Sequence of Data Elements to specify the data that is provided in the notifications for watchers changed events.

Sequence Element Name	Sequence Element Type	Notes
Event	TpPAMEventName	Contains the name of the event for which the watchers changed.
ChangeType	TpPAMwatcherChangeType	Specifies whether the listed watchers were added or deleted.
Identity	TpPAMFQName	Contains the name of the identity whose capability has changed.
Watchers	TpPAMFQNameList	Contains the list of watchers involved in the change.

### 11.12.22.3 TpPAMwatcherChangeType

This specifies the values representing the type of change that occurred to the list of watchers.

Name	Value	Description
PAM_WATCHERS_PERIODIC	0	Periodic reporting, not necessarily a change.
PAM_WATCHERS_ADDED	1	Watchers added to the list.
PAM_WATCHERS_DELETED	2	Watchers deleted from the list.



## 11.13 Error Types

### 11.13.1 TpPAMErrorCause

This defines the types of errors reported by PAM.

Name	Value	Description
P_PAM_CAUSE_UNDEFINED	0	Undefined.
P_PAM_CAUSE_INVALID_ADDRESS	1	The request cannot be handled because the address specified is not valid.
P_PAM_CAUSE_SYSTEM_FAILURE	2	System failure. The request cannot be handled because of a general problem in the service or in the underlying network.
P_PAM_CAUSE_INFO_UNAVAILABLE	3	The information is currently not available.
P_PAM_CAUSE_EVENT_REGISTRATION_CANCELLED	4	The registration for the event has been cancelled by the service.

### 11.13.2 TpPAMErrorInfo

This is a Sequence of Data Elements to specify the error notification data.

Sequence Element Name	Sequence Element Type	Description
Cause	TpPAMErrorCause	Contains information about the reason for the error.
ErrorData	TpPAMNotificationInfo	Contains information relevant to each error such as the identity for which the error exists and/or the attributes for which the error exists.

## 12 Presence and Availability Management Exception Classes

The following are the list of exception classes which are used in this interface of the API.

Name	Description
P_PAM_AGENT_EXISTS	Indicates that an Agent with the agentName already exists
P_PAM_ALIAS_EXISTS	Indicates that the specified alias is already associated to the Identity
P_PAM_ALIAS_NOT_UNIQUE	Indicates that the alias has already been assigned to another identity
P_PAM_ATTRIBUTE_EXISTS	Indicates that at least one of the named attributes already exists
P_PAM_DISASSOCIATED_TYPE	Indicates that one of the specified types is not associated with the named identity/agent
P_PAM_IDENTITY_EXISTS	Indicates that the specified Identity already exists
P_PAM_INVALID_CREDENTIAL	Indicates that the credential presented is not recognized or insufficient for the operation
P_PAM_IS_CYCLIC	Indicates that the requested operation will create cyclic relationship
P_PAM_MEMBER_EXISTS	Indicates that the specified member is already in the group
P_PAM_NO_CAPABILITY	Indicates that a supplied capability is not a capability of the requested agent. No attributes are affected
P_PAM_NOT_MEMBER	Indicates that the specified member is not member of the group
P_PAM_NOT_REGISTERED	Indicates that the interface was not previously registered
P_PAM_NOT_SUPPORTED	Implementation dependent status that indicates that this method is not supported by the implementation
P_PAM_TYPE_ASSOCIATED	Indicates that a named type has already been associated with the identity/agent
P_PAM_TYPE_EXISTS	Indicates that the named type already exists

Name	Description
P_PAM_UNASSIGNED_ALIAS	Indicates that the specified alias was not an alias of the named identity
P_PAM_UNAVAILABLE_INTERFACE	Indicates that the specified interface does not exist or is unavailable
P_PAM_UNKNOWN_AGENT	Indicates that the Agent with the specified name does not exist
P_PAM_UNKNOWN_ALIAS	Indicates that the Alias with the specified name does not exist
P_PAM_UNKNOWN_ASSIGNMENT	Indicates that no assignment exists for this identity and agent
P_PAM_UNKNOWN_ATTRIBUTE	Indicates that at least one of the specified attributes has not been defined or has not been associated with the specified object
P_PAM_UNKNOWN_ATTRIBUTES	Indicates that the specified attribute list contains attributes not part of the named object
P_PAM_UNKNOWN_CAPABILITY	Indicates that a supplied capability is not a capability of the requested agent, or has not been defined. No attributes are affected
P_PAM_UNKNOWN_GROUP	Indicates that the specified group identity does not exist
P_PAM_UNKNOWN_IDENTITY	Indicates that the specified identity does not exist
P_PAM_UNKNOWN_MEMBER	Indicates that the specified member identity does not exist
P_PAM_UNKNOWN_TYPE	Indicates that the named type does not exist / indicates that the named identity/agent has not been associated with the named type / indicates that a specified type name has not been defined as an agent type

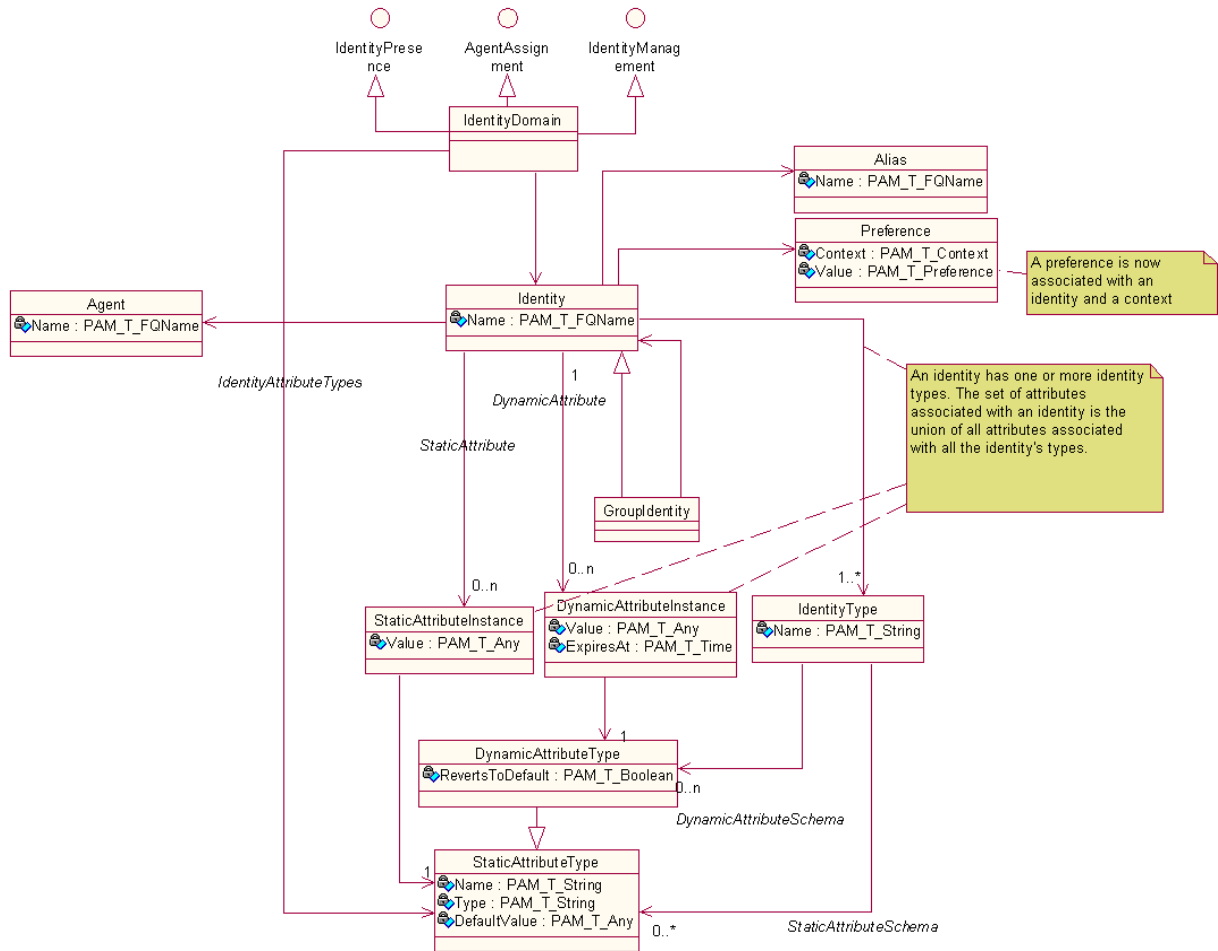
Each exception class contains the following structure.

Structure Element Name	Structure Element Type	Structure Element Description
ExtraInformation	TpString	Carries extra information to help identify the source of the exception, e.g. a parameter name.

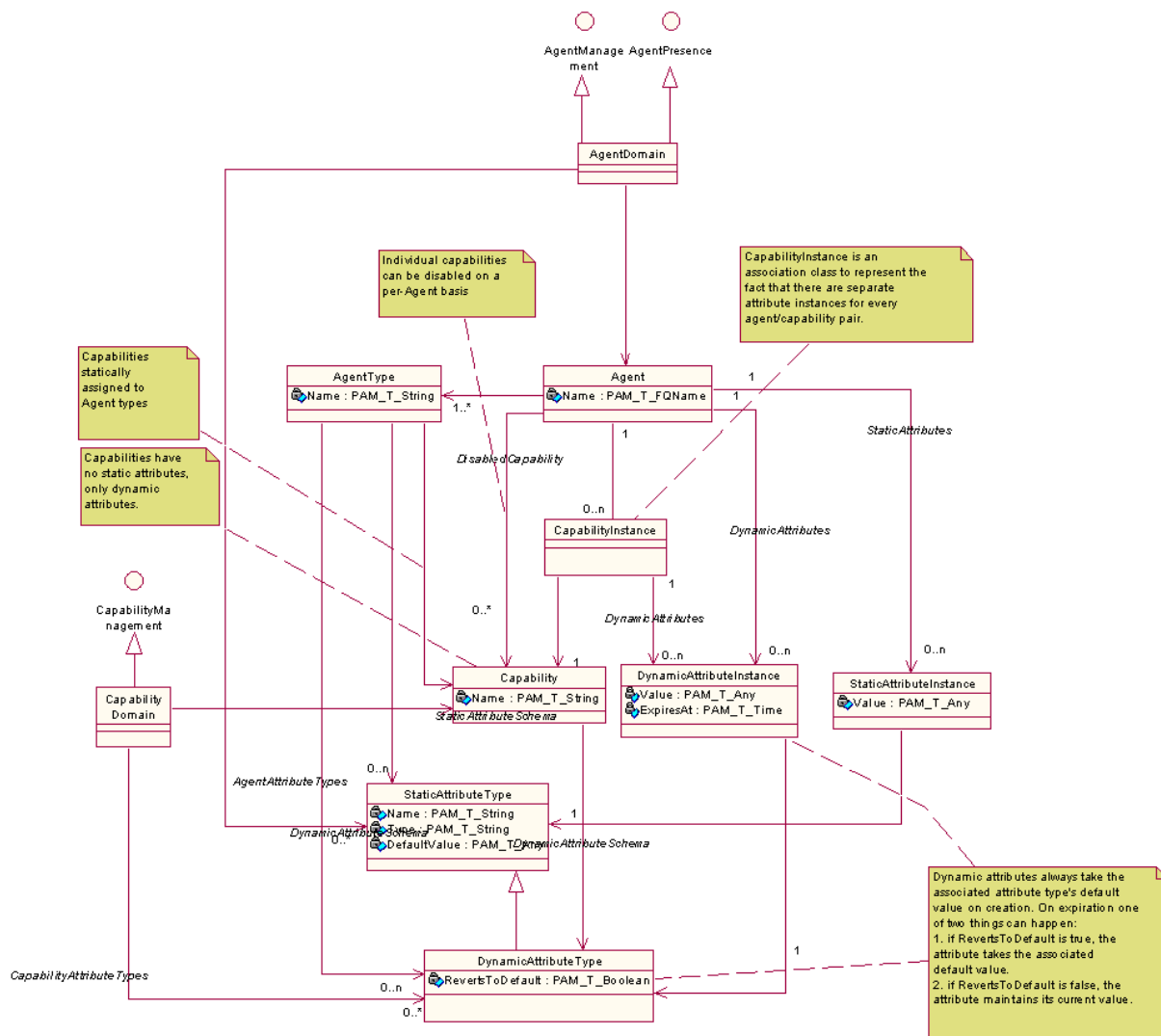
# Annex A (informative): Further PAM Information

## A.1 UML Models

### A.1.1 Identity



## A.1.2 Agent



## A.2 Model

This clause describes a suggested model for Presence and Availability Management that has influenced the design of the specifications. The model embodies assumptions about the architecture in typical usage, the security and privacy issues, the types of clients or applications that will access PAM implementations, and the framework in which they do so.

Presence and Availability Management has dual roles. In one role, it acts as an abstraction layer (figure A.1) that sits between:

- the end-users who will manage their communication identities and availability,
- the communication services that will behave according to the wishes of the end-user, and
- the communication networks to which the end-user's devices are attached and from which their status is to be obtained or inferred.

Here the goal is to "keep the end-user in the loop" i.e. to let the end-users manage their communication services as much as the service providers manage their subscribers. This role primarily determines the functionality of the specifications.

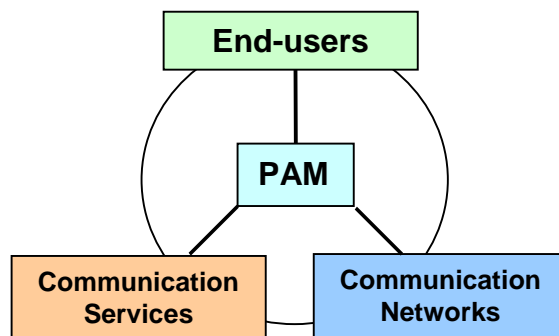


Figure A.1: "Keeping end-users in the loop"

In the second role, Presence and Availability Management provides the means for multiple administrative domains to share information about identity, presence and availability in controlled ways (figure A.2). This sharing may occur for the purposes of allowing communications between end-users in multiple domains and/or for the purposes of allowing the information to be federated into a merged global address space. In either case, this role determines the security and privacy aspects of the specification design.

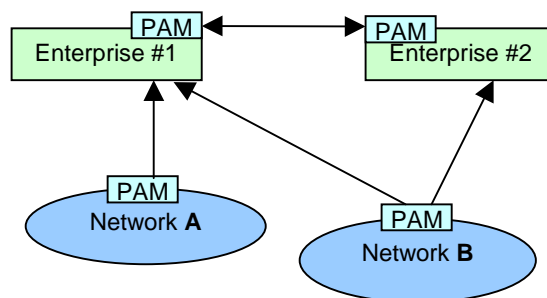


Figure A.2: Information sharing via PAM interfaces

## A.3 Architecture

There are several architectural scenarios for use of PAM depending on the participants being legacy systems or future systems or a combination of both. The least intrusive use of PAM is through an implementation of an abstraction layer over existing legacy systems to allow some limited management control for the end-user (figure A.3). The limited benefits come from being able to use third-party end-user management software written for the standard specification. The legacy systems themselves are unaware of the PAM layer. Example applications include number translation schemes in telephony systems enhanced through PAM interfaces, LDAP enabled e-mail systems extended through PAM layer to provide dynamic and/or customized address information, etc..

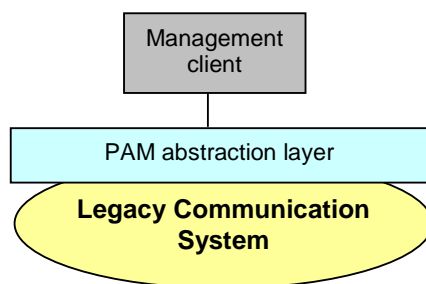
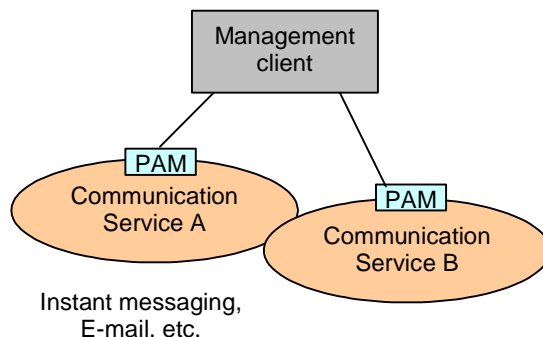


Figure A.3: Abstraction over legacy systems

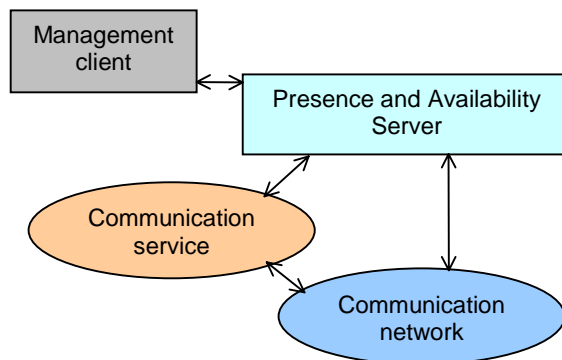
The next level of adoption comes from communication services or networks that already have some notion of identity, presence and/or availability, exporting this information through the PAM interface (figure A.4). This allows third-party end-user management software written for the standard specification to manage across multiple communication systems on behalf of the user. As the communication services are already designed to take some of these notions into account, the end-user is able to customize the services to a larger extent than the previous scenario. Example applications include Instant Messaging, Email, etc.



**Figure A.4: Exporting from communication services**

Maximum benefits from Presence and Availability Management comes in a scenario (figure A.5) in which:

- communication networks export relevant information and device status through PAM interface;
- communication services are written to consult PAM servers to affect communication handling; and
- end-user management systems written to PAM specifications allow end-users to specify policies and preferences for their communication capabilities;
- any of the communication services (voice telephony, fax, e-mail, instant messaging, etc.) can potentially use this architecture.



**Figure A.5: PAM enabled communications**

## A.4 Levels of access

As described in earlier clauses, the Presence and Availability Management platform can be used for a variety of purposes including:

- Third-party communication management software for end-users.
- Single-point administration software for enterprises.
- Federating namespaces across multiple communication services and networks.

- Exporting enterprise-managed identity, presence and/or availability data for use by external communication services/devices.
- Exporting status and/or location data of devices from networks for use by communication services.

Not all methods in every interface are likely to be used in every context. For example, a simple communication device such as an instant messaging client can check for availability of an end-user using the availability interface but may not need the preference management methods. Communication management software for the end-user may manage the preferences for the user but would not require the user identity creation methods. Enterprise administration software or services installed on top of the platform may require access to all the methods in every interface.

While the interfaces defined in the present document are a minimal union of all the potential types of accesses, it is useful to recognize three categories of platform access primarily differentiated by the authentication and security requirements as described below. Unlike other OSA specifications, PAM specification does not partition the methods and interfaces into views for each type of access. Platform implementations that implement the access framework or define the access protocols are encouraged to define views for the three types of accesses and decide on the subset of methods supported in each view. PAM specification does not require that every method be supported in implementations. Consequently, software written for the platform must take into account the possibility that any method may return with a "not supported" status.

Briefly described below are the characteristics of the three major levels of accesses.

### A.4.1 Application

Applications are independent, stand-alone software systems that access PAM implementations on a continuous basis possibly on behalf of multiple clients. Examples include PAM-compliant gateways, switches, messaging servers, address translation systems, enterprise management systems, etc. These applications are likely to be "always-on" and must be allowed to authenticate themselves only at startup of the applications. The applications are typically expected to be non-trivial and running on computing platforms that allow for heavyweight frameworks such as CORBA to be used for access to the interfaces.

### A.4.2 Service

Services are software modules that extend or provide additional functionality to PAM implementations. They may run in the same process space as the PAM platform or may run remotely. It is up to the implementations to provide implementation-specific mechanisms for adding, using or managing such services. Examples of services include "buddy list" systems, identity authentication systems, etc.

### A.4.3 Thin client

Independent software differentiated from applications in being relatively lightweight and potentially running on devices with minimal computing capabilities such as cellular phones or palm devices. The use of the PAM implementation by a thin client is expected to be infrequent and transitory. Consequently, authentication, when required, may need to happen as often as an individual method access. The PAM implementations need to provide lightweight, preferably message based access for such clients. The thin client access may not be limited to checks for presence and/or availability alone. They may provide some limited end-user preference management capabilities as well.

---

## A.5 Use cases

This clause gives some illustrative use cases of the different interfaces in the present document. This list is not meant to be an exhaustive listing of all the possible use cases but rather an aid in understanding possible uses of the PAM interfaces.

### A.5.1 Identity Management

- 1) Create an identity for John Smith (identity name) of type subscriber (identity type) with the following address information (a specific named profile for addresses).
- 2) Delete the identity for John Smith.
- 3) Create an alias of Jsmith@company.com under the namespace of "myISP" (namespace created by the ISP with whom John has an account).
- 4) Delete the alias of Jsmith@company.com for John Smith.
- 5) Create a group named "Project Manhattan" (group identity name) of type projects (identity type) with the following people in it – John, ... (group members).
- 6) Delete the group called "Project Manhattan".
- 7) Add John Smith to "Project Manhattan" Group.
- 8) Remove John Smith from "Project Manhattan" Group.
- 9) Is there a John Smith in the default namespace?
- 10) What is the address information for John Smith?
- 11) What are John Smith's aliases?
- 12) What groups does John Smith belong to?
- 13) Whose alias is Jsmith@myISP.com under the name space of "myISP"?
- 14) Who are the members of "Project Manhattan"?

### A.5.2 Agent Management

- 1) Create an agent of type "mobile phone" with the ID 123456789 (agent name) and the following capabilities - WAP, Voice, SMS.
- 2) Delete the agent with the ID 123456789.
- 3) Add the capability "video conferencing" to the agent with id 123456789.
- 4) Disable the capability "WAP" for the agent with id 123456789.
- 5) Associate the agent with ID 123456789 with the agent type "GSM phone".
- 6) Is there an agent with ID 123456789?
- 7) What agent types are associated with the agent ID 123456789?
- 8) What is Agent with ID 123456789 capable of?
- 9) Is Agent with ID 123456789 capable of SMS?



### A.5.3 Agent Assignment

- 10) John Smith now has the phone with ID 123456789.
- 11) John Smith no longer has the phone with ID 123456789.
- 12) What agents does John Smith have for instant messaging (capability)?
- 13) Is John Smith able to receive instant messages (capability)?
- 14) What all means can John Smith be communicated with?
- 15) Who is currently assigned to the phone with ID 123456789?

### A.5.4 Agent Presence

- 16) Set agent status information (dynamic attributes) for mobile phone 123456789 for Instant Messaging with the following location information valid until removed.
- 17) Set the expiration time for the location information (dynamic attribute) for mobile phone 123456789.
- 18) Update status information (a specific presence profile) of mobile phone 123456789 for Instant Messaging.
- 19) What is the call status of phone 123456789 for voice calls?
- 20) What is the location information about phone 123456789 for SMS?
- 21) What is the motion information of phone 123456789 for SMS?

### A.5.5 Identity Presence

- 22) Set status information for John Smith for Instant Messaging valid until removed.
- 23) Set call status information of John Smith for voice calls.
- 24) Set location information of John Smith for Instant Messaging.
- 25) What is the call status of John Smith for voice calls?
- 26) What is the location information about John Smith for SMS?
- 27) What is the motion information of John Smith for SMS?

### A.5.6 Availability

- 28) Store these preferences for John Smith for his voice calls.
- 29) Delete John Smith's preferences for voice calls.
- 30) What are John Smith's preferences for voice calls?
- 31) Is John Smith available to talk to Jane Doe on the phone?
- 32) How can Jane Doe contact John Smith via Instant Messaging?
- 33) What is John Smith's location information for Jane Doe?

---

## Annex B (normative): OMG IDL Description of Presence and Availability Management SCF

The OMG IDL representation of this interface specification is contained in text files (pam\_interfaces.idl and pam\_data.idl contained in archive es\_20391514v010101m0.zip) which accompany the present document.

---

## Annex C (informative): Java™ API Description of the Presence and Availability Management SCFs

The Java™ API realisation of this interface specification is produced in accordance with the Java™ Realisation rules defined in ES 203 915-1. These rules aim to deliver for Java™, a developer API, provided as a realisation, supporting a Java™ API that represents the UML specifications. The rules support the production of both J2SE™ and J2EE™ versions of the API from the common UML specifications.

The J2SE™ representation of this interface specification is provided as Java™ Code, contained in archive 20391514J2SE.zip.

The J2EE™ representation of this interface specification is provided as Java™ Code, contained in archive 20391514J2EE.zip.

Both these archives can be found in es\_20391514v010101m0.zip which accompanies the present document.

---

## Annex D (informative): Contents of 3GPP OSA R6 Presence and Availability Management

The PAM Access SCF (minus interface IpPAMAgentPresence), PAM Provisioning SCF and the Event Management SCF are relevant for TS 129 198-14 V6 (Release 6).

---

## Annex E (informative): Description of Presence and Availability Management for 3GPP2 cdma2000 networks

This annex is intended to define the OSA API Stage 3 interface definitions and it provides the complete OSA specifications. It is an extension of OSA API specifications capabilities to enable operation in cdma2000 systems environment. They are in alignment with 3GPP2 Stage 1 requirements and Stage 2 architecture defined in [52], [53] and [54] of ES 203 915-1, clause 2. These requirements are expressed as additions to and/or exclusions from the 3GPP Release 6 specification. The information given here is to be used by developers in 3GPP2 cdma2000 network architecture to interpret the 3GPP OSA specifications.

---

### E.1 General Exceptions

The term UMTS is not applicable for the cdma2000 family of standards. Nevertheless these terms are used (TR 121 905) mostly in the broader sense of "3G Wireless System". If not stated otherwise there are no additions or exclusions required.

CAMEL and CAP mappings are not applicable for cdma2000 systems.

---

### E.2 Specific Exceptions

#### E.2.1 Clause 1: Scope

There are no additions or exclusions.

#### E.2.2 Clause 2: References

Normative references on TS 123 078 and on TS 129 078 are not applicable for cdma2000 systems.

#### E.2.3 Clause 3: Definitions and abbreviations

There are no additions or exclusions.

#### E.2.4 Clause 4: Presence and Availability Management SCF

There are no additions or exclusions.

#### E.2.5 Clause 5: Sequence Diagrams

There are no additions or exclusions.

#### E.2.6 Clause 6 Class Diagrams

There are no additions or exclusions.

#### E.2.7 Clause 7: The Service Interface Specifications

There are no additions or exclusions.

## **E.2.8 Clause 8: Presence and Availability Management Interface Classes**

There are no additions or exclusions.

## **E.2.9 Clause 9: State Transition Diagrams**

There are no additions or exclusions.

## **E.2.10 Clause 10: PAM Service Properties**

There are no additions or exclusions.

## **E.2.11 Clause 11: PAM Data Definitions**

There are no additions or exclusions.

## **E.2.12 Clause 12: Presence and Availability Management Exception Classes**

There are no additions or exclusions.

## **E.2.13 Annex B (normative): OMG IDL Description of Presence and Availability Management SCF**

There are no additions or exclusions.

## **E.2.14 Annex C (informative): Java™ API Description of Presence and Availability Management SCF**

There are no additions or exclusions.

---

## Annex F (informative): Record of changes

The following is a list of the changes made to the present document for each release. The list contains the names of all changed, deprecated, added or removed items in the specifications and not the actual changes. Any type of change information that is important to the reader is put in the final clause of this annex.

Changes are specified as changes to the prior major release, but every minor release will have its own part of the table allowing the reader to know when the actual change was made.

---

### F.1 Interfaces

#### F.1.1 New

Identifier	Comments
<b>Interfaces added in ES 203 915-14 version 1.1.1 (Parlay 5.0)</b>	

#### F.1.2 Deprecated

Identifier	Comments
<b>Interfaces deprecated in ES 203 915-14 version 1.1.1 (Parlay 5.0)</b>	

#### F.1.3 Modified

Identifier	Comments
<b>Interfaces modified in ES 203 915-14 version 1.1.1 (Parlay 5.0)</b>	

#### F.1.4 Removed

Identifier	Comments
<b>Interfaces removed in ES 203 915-14 version 1.1.1 (Parlay 5.0)</b>	

---

## F.2 Methods

### F.2.1 New

Identifier	Comments
<b>Methods added in ES 203 915-14 version 1.1.1 (Parlay 5.0)</b>	
IpPAMPresenceAvailabilityManager.activateService	
IpPAMPresenceAvailabilityManager.deactivateService	
IpPAMPresenceAvailabilityManager.isActiveIdentity	
IpPAMEventManager.activateService	
IpPAMEventManager.deactivateService	
IpPAMEventManager.isActiveIdentity	

### F.2.2 Deprecated

Identifier	Comments
<b>Methods deprecated in ES 203 915-14 version 1.1.1 (Parlay 5.0)</b>	

### F.2.3 Modified

Identifier	Comments
<b>Methods modified in ES 203 915-14 version 1.1.1 (Parlay 5.0)</b>	

### F.2.4 Removed

Identifier	Comments
<b>Methods removed in ES 203 915-14 version 1.1.1 (Parlay 5.0)</b>	

---

## F.3 Data Definitions

### F.3.1 New

Identifier	Comments
<b>Data Definitions added in ES 203 915-14 version 1.1.1 (Parlay 5.0)</b>	

### F.3.2 Modified

Identifier	Comments
<b>Data Definitions modified in ES 203 915-14 version 1.1.1 (Parlay 5.0)</b>	



### F.3.3 Removed

Identifier	Comments
Data Definitions removed in ES 203 915-14 version 1.1.1 (Parlay 5.0)	

---

## F.4 Service Properties

### F.4.1 New

Identifier	Comments
Service Properties added in ES 203 915-14 version 1.1.1 (Parlay 5.0)	

### F.4.2 Deprecated

Identifier	Comments
Service Properties deprecated in ES 203 915-14 version 1.1.1 (Parlay 5.0)	

### F.4.3 Modified

Identifier	Comments
Service Properties modified in ES 203 915-14 version 1.1.1 (Parlay 5.0)	

### F.4.4 Removed

Identifier	Comments
Service Properties removed in ES 203 915-14 version 1.1.1 (Parlay 5.0)	

---

## F.5 Exceptions

### F.5.1 New

Identifier	Comments
Exceptions added in ES 203 915-14 version 1.1.1 (Parlay 5.0)	

## F.5.2 Modified

Identifier	Comments
Exceptions modified in ES 203 915-14 version 1.1.1 (Parlay 5.0)	

## F.5.3 Removed

Identifier	Comments
Exceptions removed in ES 203 915-14 version 1.1.1 (Parlay 5.0)	

---

## F.6 Others

None.

---

## History

<b>Document history</b>		
V1.1.1	February 2005	Membership Approval Procedure MV 20050408: 2005-02-08 to 2005-04-08