Final draft ETSI ES 201 915-9 V1.2.1 (2002-05)

ETSI Standard

Open Service Access (OSA);
Application Programming Interface (API);
Part 9: Generic Messaging SCF

Reference

RES/SPAN-120076-9

Keywords

API, IDL, UML, OSA

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

Contents

Intellectual Property Rights

Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Services and Protocols for Advanced Networks (SPAN), and is now submitted for the ETSI standards Membership Approval Procedure.

The present document is part 9 of a multi-part deliverable covering Open Service Access (OSA); Application Programming Interface (API), as identified below. The API specification (ES 201 915) is structured in the following parts:

Part 1:	"Overview";
Part 2:	"Common Data Definitions";
Part 3:	"Framework";
Part 4:	"Call Control SCF";
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".

The present document has been defined jointly between ETSI, The Parlay Group [24] of ES 201 915-1 and the 3GPP, in co-operation with a number of JAIN™ Community [25] of ES 201 915-1 member companies.

The present document forms part of the Parlay 3.1 set of specifications.

1 Scope

The present document is part 9 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 Generic Messaging Service Capability Feature (SCF) aspects of the interface. All aspects of the Generic Messaging 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 201 915-1 contain provisions which, through reference in this text, constitute provisions of the present document.

ETSI ES 201 915-1: "Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ES 201 915-1 apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations defined in ES 201 915-1 apply.

4 Generic Messaging SCF

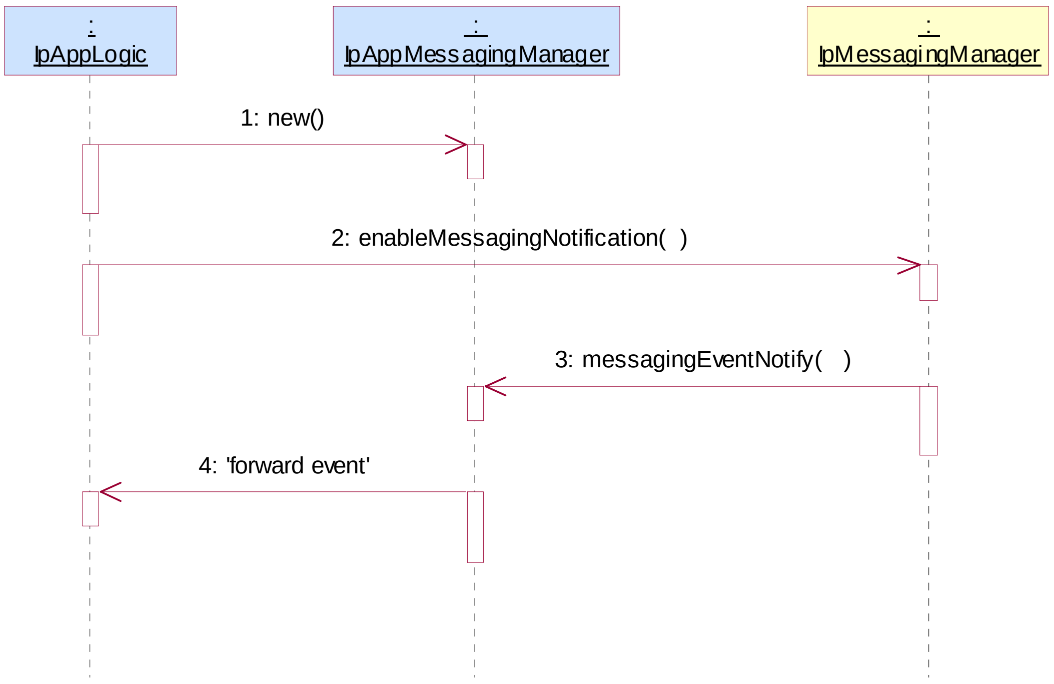
The following clauses describe each aspect of the Generic Messaging Service Capability Feature (SCF).

The order is as follows:

- The Sequence diagrams give the reader a practical idea of how each of the SCF is implemented.
- The Class relationships clause show how each of the interfaces applicable to the SCF, relate to one another.
- The Interface specification clause describes in detail each of the interfaces shown within the Class diagram part.
- The State Transition Diagrams (STD) show the transition between states in the SCF. The states and transitions are well-defined; either methods specified in the Interface specification or events occurring in the underlying networks cause state transitions.
- The Data Definitions clause show a detailed expansion of each of the data types associated with the methods within the classes. Note that some data types are used in other methods and classes and are therefore defined within the Common Data types part of this specification.

5 Sequence Diagrams

5.1 Prepare Mailbox



1: This message is used by the application to create an object implementing the IpAppMessagingManager interface.

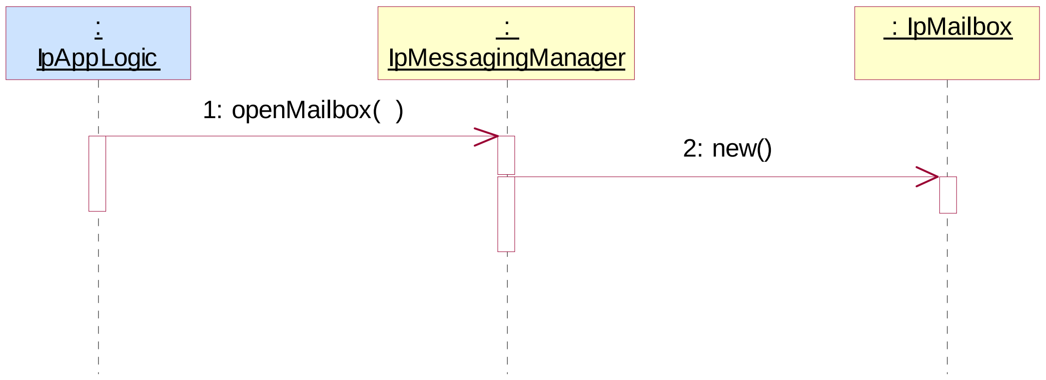
2: This message is used to enable the notification mechanism so that events can be sent to the application.

When new mail, that matches the event criteria set in message 2, arrives a message indicating the presence of new mail (not shown) is directed to the object implementing the IpMessagingManager.

3: This message is used to pass the new mail event to the object implementing the IpAppMessagingManager interface.

4: This message is used to forward message 3 to the IpAppLogic.

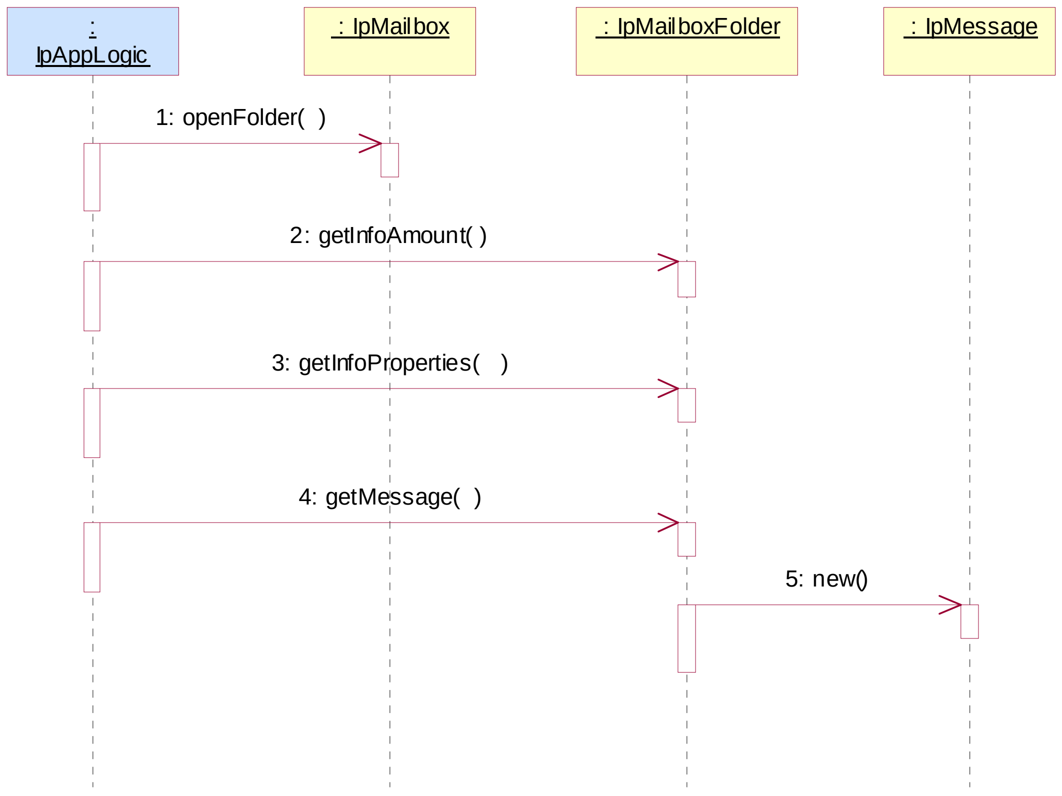
5.2 Open Mailbox



1: This message requests the object implementing the IpMessagingManager interface to create an object implementing the IpMailbox interface.

2: Assuming that the criteria for creating an object implementing the IpMailbox interface is met, message 2 is used to create it.

5.3 Get Message



1: This message requests a folder to be opened and returns a reference to that folder.

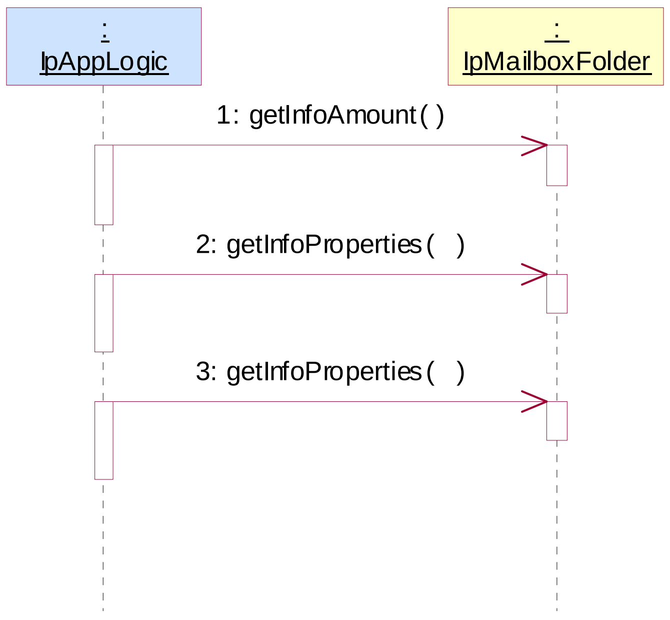
2: This message requests the number of folder information properties of the opened folder.

3: This message requests all of the folder information properties.

4: This message requests a message from the opened mailbox folder.

5: Assuming that the criteria for creating an object implementing the IpMessage interface are met, the (internal) message 5 is used to create it.

5.4 Get Folder Information

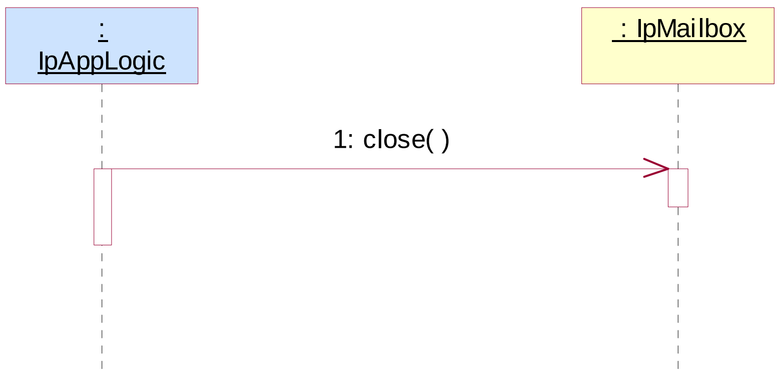


1: This message requests the number of folder information properties of the specified folder.

2: This message requests the first set of folder information properties.

3: This message requests the second set of folder information properties.

5.5 Close Mailbox



1: This message requests the object implementing the IpMailbox interface to de-assign.

6 Class Diagrams

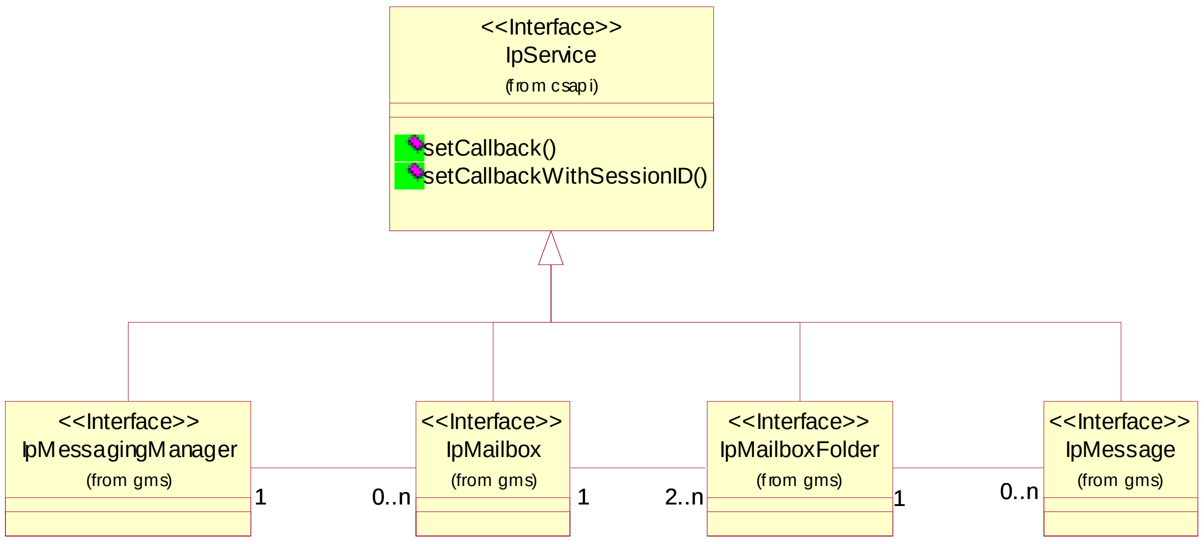


Figure 1: Package Overview: Service Interfaces

The application generic messaging service package consists of only one IpAppMessagingManager interface.

The generic messaging service package consists of one IpMessagingManager interface, zero or more IpMailbox interfaces, zero or more IpMailboxFolder and zero or more IpMessage interfaces.

The class diagram in the following figure shows the interfaces that make up the application generic messaging service package and the generic messaging service package. Communication between these packages is done via the +uses the IpMessagingManager channels. Communication with the IpMailbox and IpMailboxFolder interfaces has to be done via the application logic (not shown).

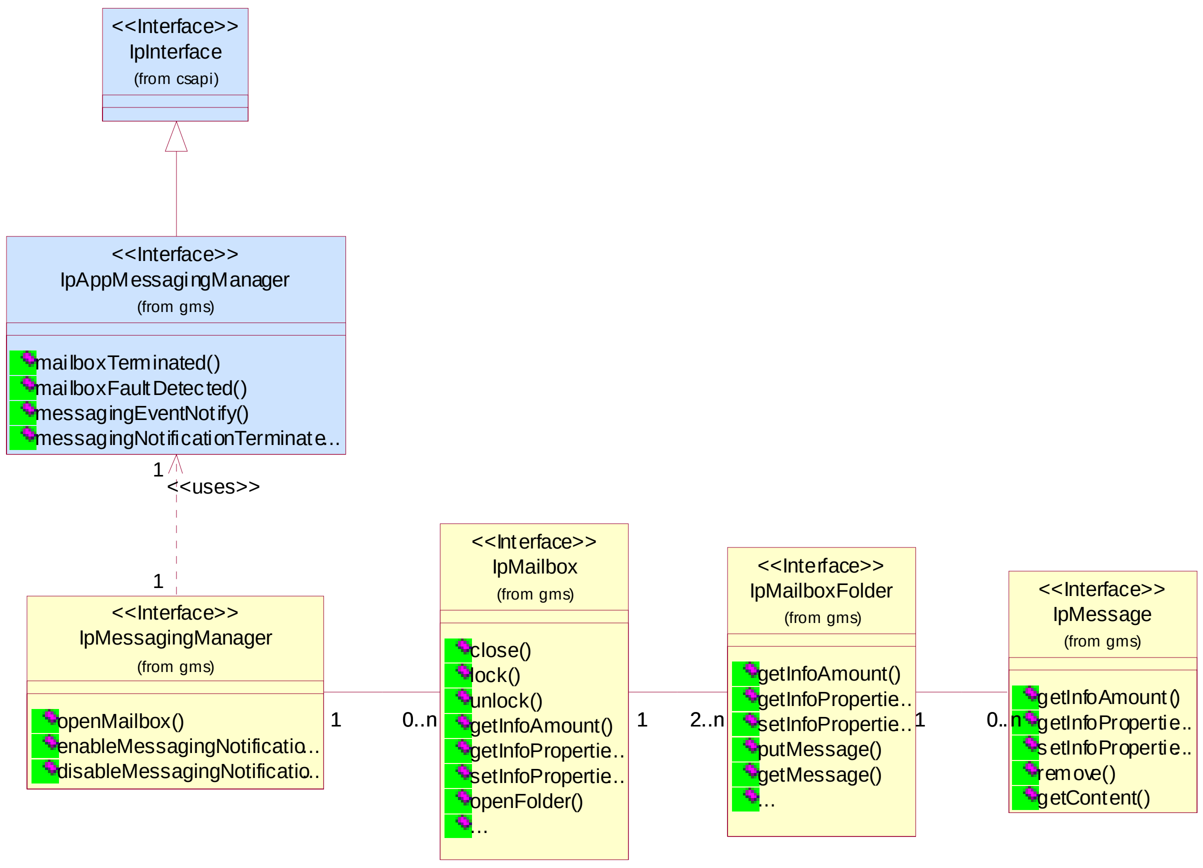


Figure 2: Package Overview: Application and Service Interfaces

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 in clauses 7.1.1 to 7.1.4.

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.

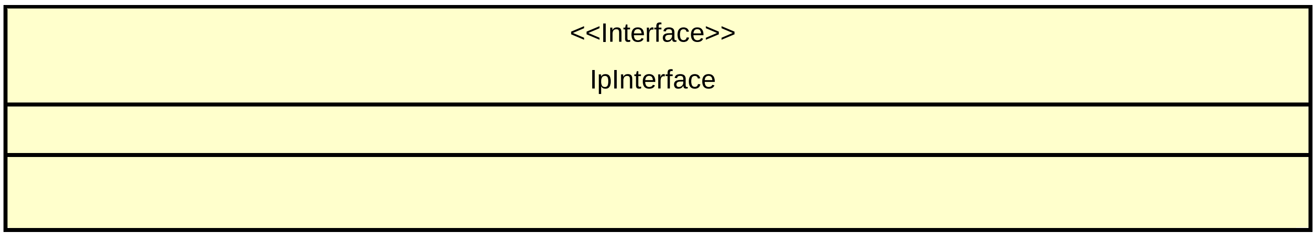
7.1.4 State Model

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.

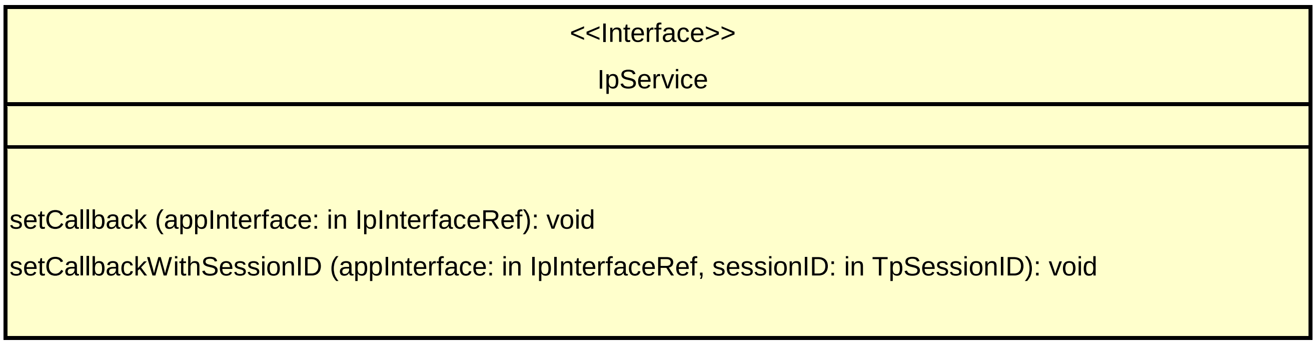
The interfaces that are implemented by the services are denoted as "Service Interface". The corresponding interfaces that must be implemented by the application (e.g. for API callbacks) are denoted as "Application Interface".

7.4 Generic Service Interface

7.4.1 Interface Class IpService

Inherits from: IpInterface

All service interfaces inherit from the following interface.



<<Interface>>

IpService

setCallback (appInterface: in IpInterfaceRef): void

setCallbackWithSessionID (appInterface: in IpInterfaceRef, sessionID: in TpSessionID): void

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.

Parameters

`appInterface: in IpInterfaceRef`

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

Raises

`TpCommonExceptions, P_INVALID_INTERFACE_TYPE`

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.

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 Generic Messaging Interface Classes

The Generic Messaging Service interface (GMS) is used by applications to send, store and receive messages. GMS has voice mail and electronic mail as the messaging mechanisms. The messaging service interface can be used by both.

A messaging system is assumed to have the following entities:

Mailboxes. This is the application's main entry point to the messaging system. The framework may or may not need to authenticate an application before it accesses a mailbox

Folders. A mailbox has at least the inbox and the outbox as folders. The name of the inbox is "INBOX", and the name of the outbox is "OUTBOX". These folders may have sub-folders. The names of these sub-folders are appended to their parents' names with "/" as the delimiter. For instance, if there is a sub-folder in INBOX called "Personal" and a sub-folder in that folder called "archive" then the fully qualified names, which are required for all operations, of the four folders are "INBOX", "OUTBOX", "INBOX/Personal", and "INBOX/Personal/archive". The names are case sensitive. A messaging service may have other folders other than the inbox and the outbox.

Messages. Messages are stored in folders. Messages usually have properties associated with them.

The GMS is represented by the IpMessagingManager, IpMailbox, IpMailboxFolder and IpMessage interfaces to services provided by the network. To handle responses and reports, the developer must implement IpAppMessagingManager to provide the callback mechanism for the Messaging service manager.

8.1 Interface Class IpMessagingManager

Inherits from: IpService.

This interface is the "service manager" interface for the Generic Messaging Service. The generic messaging manager interface provides the management functions to the generic messaging service. The application programmer can use this interface to open mailbox objects and also to enable or disable event notifications.

<<Interface>> IpMessagingManager
openMailbox (mailboxID: in TpAddress, authenticationInfo: in TpString): TpMailboxIdentifier enableMessagingNotification (appInterface: in IpAppMessagingManagerRef, eventCriteria: in TpMessagingEventCriteria): TpAssignmentID disableMessagingNotification (assignmentID: in TpAssignmentID): void

Method
`openMailbox()`

This method opens a mailbox for the application. The session ID for use by the application is returned. Authentication information may be needed to open the mailbox.

The application can open more than one mailbox at the same time. The application is not allowed to open the same mailbox more than once at the same time.

Returns: mailboxReference

Specifies the reference to the opened mailbox.

Parameters

`mailboxID: in TpAddress`

Specifies the identity of the mailbox. If the mailbox chosen is invalid, the error code P_GMS_INVALID_MAILBOX is returned.

`authenticationInfo: in TpString`

Authentication information needed for the application to open a mailbox in the messaging system, such as a key or password. If the authentication process is considered strong enough for the application to gain access to the mailbox, then the authentication information will be null. If the authentication information is not valid, the error code P_GMS_INVALID_AUTHENTICATION_INFORMATION is returned.

Returns

`TpMailboxIdentifier`

Raises

`TpCommonExceptions,P_GMS_INVALID_MAILBOX,P_GMS_INVALID_AUTHENTICATION_INFORMATION`

Method
`enableMessagingNotification()`

This method is used to enable messaging notifications so that events can be sent to the application.

Returns: assignmentID

Specifies the ID assigned by the generic messaging manager interface for this newly-enabled event notification.

Parameters

`appInterface: in IpAppMessagingManagerRef`

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface, which is used for callbacks. If set to NULL, the application interface defaults to the interface specified via the setCallback() method.

`eventCriteria: in TpMessagingEventCriteria`

Specifies the event specific criteria used by the application to define the event required.

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions, P_INVALID_CRITERIA`

Method
`disableMessagingNotification()`

This method is used by the application to disable call notifications.

Parameters

`assignmentID: in TpAssignmentID`

Specifies the assignment ID given by the generic messaging manager interface when the previous enableNotification() was called. If the assignment ID does not correspond to one of the valid assignment IDs, the framework will return the error code P_INVALID_ASSIGNMENTID.

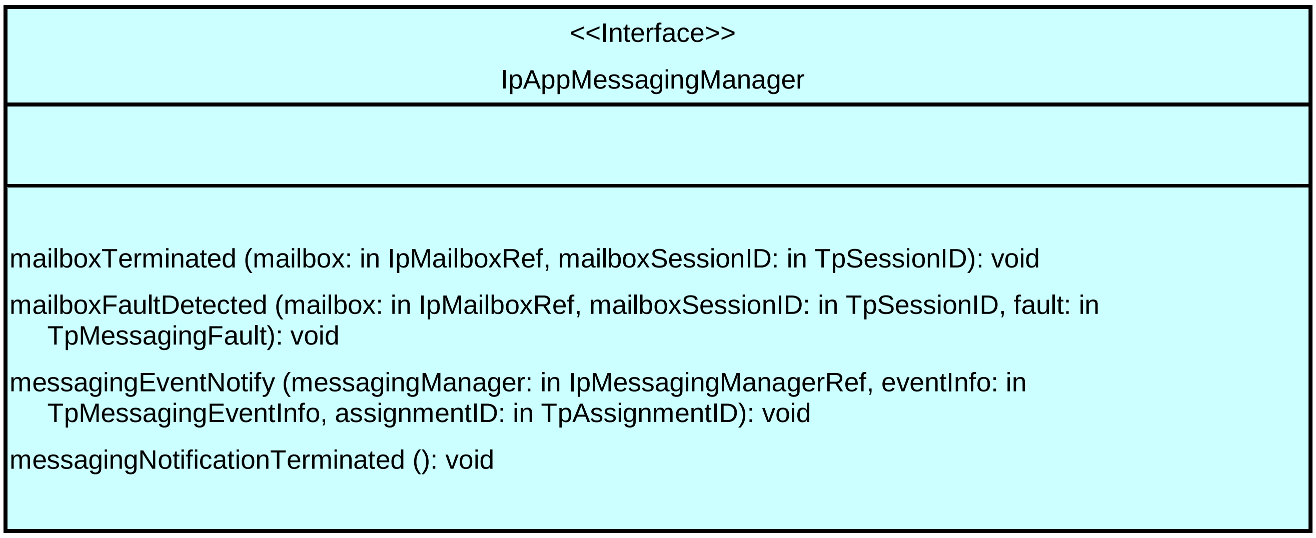
Raises

`TpCommonExceptions, P_INVALID_ASSIGNMENT_ID`

8.2 Interface Class IpAppMessagingManager

Inherits from: IpInterface.

The client application developer implements the generic messaging manager application interface to handle mailbox termination, mailbox fault and messaging event notifications.



<<Interface>>

IpAppMessagingManager

mailboxTerminated (mailbox: in IpMailboxRef, mailboxSessionID: in TpSessionID): void

mailboxFaultDetected (mailbox: in IpMailboxRef, mailboxSessionID: in TpSessionID, fault: in TpMessagingFault): void

messagingEventNotify (messagingManager: in IpMessagingManagerRef, eventInfo: in TpMessagingEventInfo, assignmentID: in TpAssignmentID): void

messagingNotificationTerminated (): void

Method
`mailboxTerminated()`

This method indicates to the application that the mailbox has terminated or closed abnormally. No further communication will be possible between the mailbox and application.

Parameters

`mailbox: in IpMailboxRef`

Specifies the interface of the mailbox that has terminated.

`mailboxSessionID: in TpSessionID`

Specifies the mailbox session ID of the mailbox that has terminated.

Method
`mailboxFaultDetected()`

This method indicates to the application that a fault has been detected in the mailbox.

Parameters

`mailbox: in IpMailboxRef`

Specifies the interface of the mailbox in which the fault has been detected.

`mailboxSessionID: in TpSessionID`

Specifies the mailbox session ID of the mailbox in which the fault has been detected.

`fault: in TpMessagingFault`

Specifies the fault that has been detected.

Method
`messagingEventNotify()`

This method notifies the application of the arrival of a messaging-related event.

Parameters

`messagingManager: in IpMessagingManagerRef`

Specifies the reference to the messaging manager interface to which the notification relates.

`eventInfo: in TpMessagingEventInfo`

Specifies data associated with this event.

`assignmentID: in TpAssignmentID`

Specifies the assignment id which was returned by the enableNotification() method. The application can use assignment id to associate events with event specific criteria and to act accordingly.

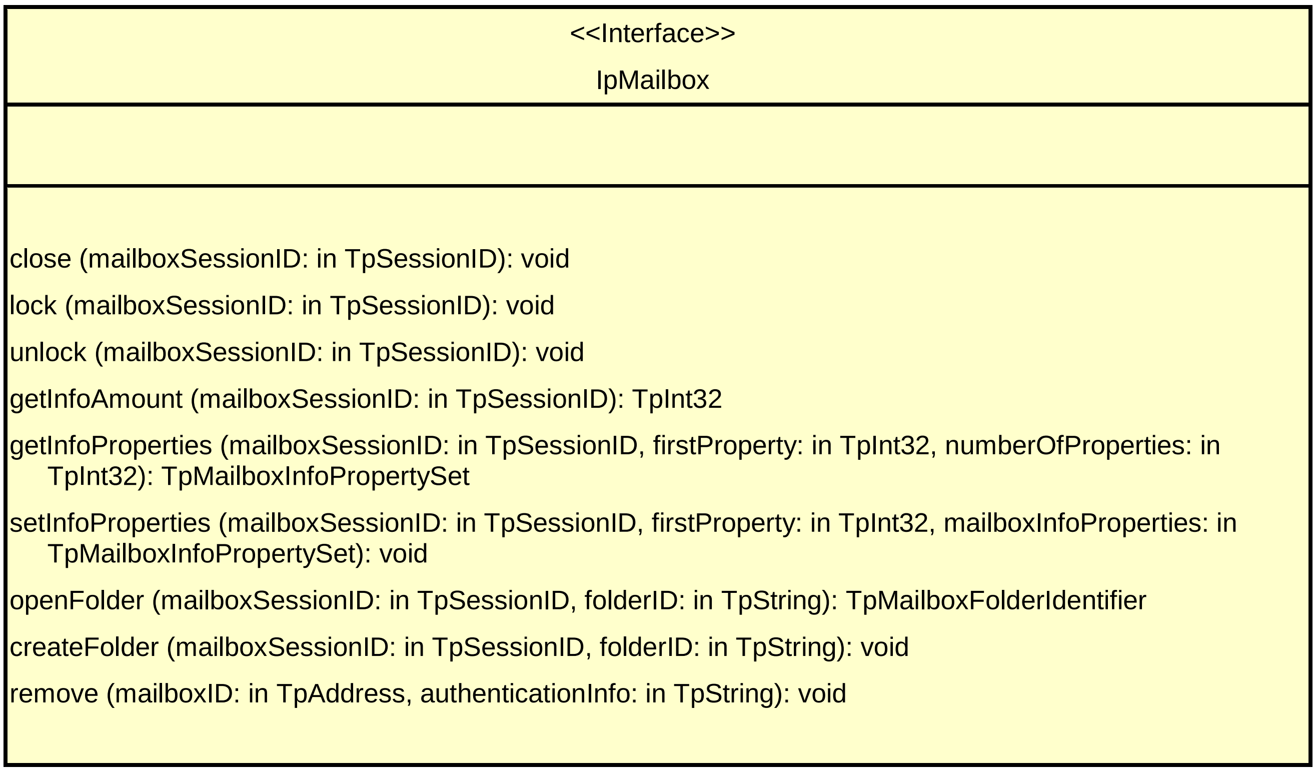
Method
`messagingNotificationTerminated()`

This method indicates to the application that all event notifications have been terminated (for example, due to faults detected).

Parameters
No Parameters were identified for this method.

8.3 Interface Class IpMailbox

Inherits from: IpService.



<<Interface>>

IpMailbox

close (mailboxSessionID: in TpSessionID): void

lock (mailboxSessionID: in TpSessionID): void

unlock (mailboxSessionID: in TpSessionID): void

getInfoAmount (mailboxSessionID: in TpSessionID): TpInt32

getInfoProperties (mailboxSessionID: in TpSessionID, firstProperty: in TpInt32, numberOfProperties: in TpInt32): TpMailboxInfoPropertySet

setInfoProperties (mailboxSessionID: in TpSessionID, firstProperty: in TpInt32, mailboxInfoProperties: in TpMailboxInfoPropertySet): void

openFolder (mailboxSessionID: in TpSessionID, folderID: in TpString): TpMailboxFolderIdentifier

createFolder (mailboxSessionID: in TpSessionID, folderID: in TpString): void

remove (mailboxID: in TpAddress, authenticationInfo: in TpString): void

Method
`close()`

This method closes the mailbox. After closing, the interfaces to the mailbox and any associated folders are automatically de-assigned and are no longer valid. Any open folders will also be automatically closed.

Parameters

`mailboxSessionID: in TpSessionID`

The session ID of the open mailbox previously opened by openMailbox. From now on, the session ID is no longer valid. If by coincidence an identical session ID is returned by a subsequent openMailbox, the session ID will be associated with the new session and has nothing to do with the closed session. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID`

Method
`lock()`

This method locks the mailbox so that only the requesting application can have access to this mailbox. Updates to the mailbox by other applications or the network are not permitted until the mailbox has been unlocked - attempts to do so result in the error code P_GMS_MAILBOX_LOCKED. When the application exits, however, all mailboxes locked by the application are unlocked automatically.

The service returns an error code P_GMS_LOCKING_LOCKED_MAILBOX when the application attempts to lock a mailbox that is locked.

Parameters

`mailboxSessionID: in TpSessionID`

This is the session ID of the open mailbox. If the session ID is not valid, the error code P_INVALID_SESSION_ID is returned.

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID, P_GMS_LOCKING_LOCKED_MAILBOX`

Method
`unlock()`

This method unlocks a previously locked mailbox. An error is returned if the mailbox is already unlocked.

Parameters

`mailboxSessionID: in TpSessionID`

This is the session ID of the locked mailbox. If the sessionID does not correspond to a locked mailbox, the error code P_GMS_UNLOCKING_UNLOCKED_MAILBOX is returned. If the application attempts to unlock a mailbox that is already locked by another application, the error code P_GMS_CANNOT_UNLOCK_MAILBOX is returned.

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID,`
`P_GMS_UNLOCKING_UNLOCKED_MAILBOX, P_GMS_CANNOT_UNLOCK_MAILBOX`

Method
`getInfoAmount()`

This method returns the number of mailbox information properties of the specified mailbox.

Returns: numberOfProperties

The number of properties associated with the folder. The number of properties is zero or positive.

Parameters

`mailboxSessionID: in TpSessionID`

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

Returns

`TpInt32`

Raises

`TpCommonExceptions,P_INVALID_SESSION_ID`

Method
`getInfoProperties()`

This method returns the properties of a mailbox.

Returns: mailboxInfoProperties

The mailbox information properties (names and values) present in the folder.

Parameters

`mailboxSessionID: in TpSessionID`

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

`firstProperty: in TpInt32`

This is the first property of interest. This number represents the starting point where the first property of the list to be retrieved from the mailbox is located. Properties are numbered from zero.

`numberOfProperties: in TpInt32`

The number of properties to return. If the value of this parameter is zero, then all properties will be returned. Otherwise, the value must be a positive number. If the number is not positive, the error code P_GMS_NUMBER_NOT_POSITIVE is returned.

Returns

`TpMailboxInfoPropertySet`

Raises

`TpCommonExceptions,P_INVALID_SESSION_ID,P_GMS_NUMBER_NOT_POSITIVE`

Method
`setInfoProperties()`

Sets the properties of a mailbox.

Parameters

`mailboxSessionID: in TpSessionID`

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

`firstProperty: in TpInt32`

This is the first property of interest. This number represents the starting point where the first property of the list to be updated in the mailbox is located. Properties are numbered from zero.

`mailboxInfoProperties: in TpMailboxInfoPropertySet`

This specifies the mailbox information properties (names and values) to be set in the mailbox. If the properties cannot be changed, then the error code P_GMS_PROPERTY_NOT_SET is returned.

Raises

`TpCommonExceptions,P_INVALID_SESSION_ID,P_GMS_PROPERTY_NOT_SET,P_GMS_MAILBOX_LOCKED`

Method

`openFolder()`

This method opens a folder for the application, and returns a folder session ID and a reference to the interface of the folder opened.

The application can open more than one folder at the same time. The application is not allowed to open the same folder more than once at the same time. If the folder is already open, the error code P_GMS_FOLDER_IS_OPEN is returned.

Returns: folderReference

Specifies the reference to the opened folder.

Parameters

`mailboxSessionID: in TpSessionID`

This is the session ID of the open mailbox.

`folderID: in TpString`

Specifies the identity of the folder. If the folder ID given is not present, the error code P_GMS_INVALID_FOLDER_ID is returned.

Returns

`TpMailboxFolderIdentifier`

Raises

`TpCommonExceptions,P_INVALID_SESSION_ID,P_GMS_FOLDER_IS_OPEN,P_GMS_INVALID_FOLDER_ID,P_GMS_MAILBOX_LOCKED`

Method

`createFolder()`

This method creates a new folder in the mailbox.

Parameters

`mailboxSessionID: in TpSessionID`

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

`folderID: in TpString`

Specifies the identity of the folder. If the folder ID given is already present, the error code P_GMS_INVALID_FOLDER_ID is returned.

Raises

`TpCommonExceptions,P_INVALID_SESSION_ID,P_GMS_INVALID_FOLDER_ID,P_GMS_MAILBOX_LOCKED`

Method

`remove()`

This method removes a mailbox from the messaging system for the application. Authentication information may be needed to remove the mailbox. If the application does not have sufficient privilege to remove the mailbox, the error code P_GMS_INSUFFICIENT_PRIVILEGE is returned.

Parameters

`mailboxID: in TpAddress`

Specifies the identity of the mailbox. If the mailbox chosen is invalid, the error code P_GMS_INVALID_MAILBOX is returned. If the mailbox is locked then the error code P_GMS_MAILBOX_LOCKED is returned. If the mailbox is open then the error code P_GMS_MAILBOX_OPEN is returned.

`authenticationInfo: in TpString`

Authentication information needed for the application to remove a mailbox from the messaging system, such as a key or password. If the authentication process is considered strong enough for the application to gain access to the mailbox, then the authentication information will be null. If the authentication information is not valid, the error code P_GMS_INVALID_AUTHENTICATION_INFORMATION is returned.

Raises

`TpCommonExceptions,P_GMS_INSUFFICIENT_PRIVILEGE,P_GMS_INVALID_MAILBOX,P_GMS_MAILBOX_LOCKED,P_GMS_MAILBOX_OPEN,P_GMS_INVALID_AUTHENTICATION_INFORMATION`

8.4 Interface Class IpMailboxFolder

Inherits from: IpService.

<<Interface>> IpMailboxFolder
getInfoAmount (folderSessionID: in TpSessionID): TpInt32 getInfoProperties (folderSessionID: in TpSessionID, firstProperty: in TpInt32, numberOfProperties: in TpInt32): TpFolderInfoPropertySet setInfoProperties (folderSessionID: in TpSessionID, firstProperty: in TpInt32, folderInfoProperties: in TpFolderInfoPropertySet): void putMessage (folderSessionID: in TpSessionID, message: in TpMessage, messageInfoProperties: in TpMessageInfoPropertySet): void getMessage (folderSessionID: in TpSessionID, messageID: in TpString): IpMessageRef close (mailboxSessionID: in TpSessionID, folderSessionID: in TpSessionID): void remove (mailboxSessionID: in TpSessionID, folderID: in TpString): void

Method
getInfoAmount()

This method returns the number of folder information properties of the specified folder.

Returns: numberOfProperties

The number of properties associated with the folder. The number of properties is zero or positive.

Parameters

folderSessionID: in TpSessionID

This is the session ID of the open folder. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

Returns

TpInt32

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method
getInfoProperties()

This method returns the properties of a folder.

Returns: folderInfoProperties

The folder information properties (names and values) present in the folder. Folder properties include parent folder, sub folders, number of messages contained, date created, date last accessed, and read/write access.

Parameters

`folderSessionID: in TpSessionID`

This is the session ID of the open folder. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

`firstProperty: in TpInt32`

This is the first property of interest. This number represents the starting point where the first property of the list to be retrieved from the folder is located. Properties are numbered from zero.

`numberOfProperties: in TpInt32`

The number of properties to return. If the value of this parameter is zero, then all properties will be returned. Otherwise, the value must be a positive number. If the number is not positive, the error code P_GMS_NUMBER_NOT_POSITIVE is returned.

Returns

`TpFolderInfoPropertySet`

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID, P_GMS_NUMBER_NOT_POSITIVE`

Method
`setInfoProperties()`

Sets the properties of a folder.

Parameters

`folderSessionID: in TpSessionID`

This is the session ID of the open folder. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

`firstProperty: in TpInt32`

This is the first property of interest. This number represents the starting point where the first property of the list to be updated in the folder is located. Properties are numbered from zero.

`folderInfoProperties: in TpFolderInfoPropertySet`

This specifies the folder information properties (names and values) to be set in the folder. Folder properties that may be changed include parent folder, sub folders and read/write access. If the properties cannot be changed, then the error code P_GMS_PROPERTY_NOT_SET is returned.

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID, P_GMS_PROPERTY_NOT_SET`

Method
putMessage()

This method puts a message into an open mailbox folder. The message and the headers are transferred to the Messaging service. The message will be taken as is. No checking is done on the message. Further more, the message is assumed to be a simple message, that is, with no attachments. If the application knows the messaging system and understands the format to send attachments, it can do so. The service will not flag any inconsistencies if the formatting of the message is not correct.

Parameters

folderSessionID: in TpSessionID

This is the session ID of the open folder. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

message: in TpMessage

The message to put into the mailbox.

messageInfoProperties: in TpMessageInfoPropertySet

This specifies the message information properties (names and values).

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method
getMessage()

This method gets a message from an open mailbox folder. The message ID can be obtained by calling the getFolderInfo and getFolderInfoProperties or embedded in an event notification from the messaging service, with information on the mailbox and notifications contained in that operation.

Returns: message

The message associated with the messageID.

Parameters

folderSessionID: in TpSessionID

This is the session ID of the open folder. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

messageID: in TpString

Specifies the identity of the message. If the message ID given is not present, the error code P_GMS_INVALID_MESSAGE_ID is returned.

Returns

IpMessageRef

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_GMS_INVALID_MESSAGE_ID

Method
`close()`

This method closes a specified folder. All subfolders of the folder are also closed.

Parameters

`mailboxSessionID: in TpSessionID`

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

`folderSessionID: in TpSessionID`

Specifies the folder session ID of the folder to close.

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID`

Method
`remove()`

This method removes a folder from the mailbox. All subfolders of the folder are also removed. The folder must be already closed, otherwise the error code P_GMS_FOLDER_IS_OPEN is returned. If the application does not have sufficient privilege to remove the folder, the error code P_GMS_INSUFFICIENT_PRIVILEGE is returned.

Parameters

`mailboxSessionID: in TpSessionID`

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

`folderID: in TpString`

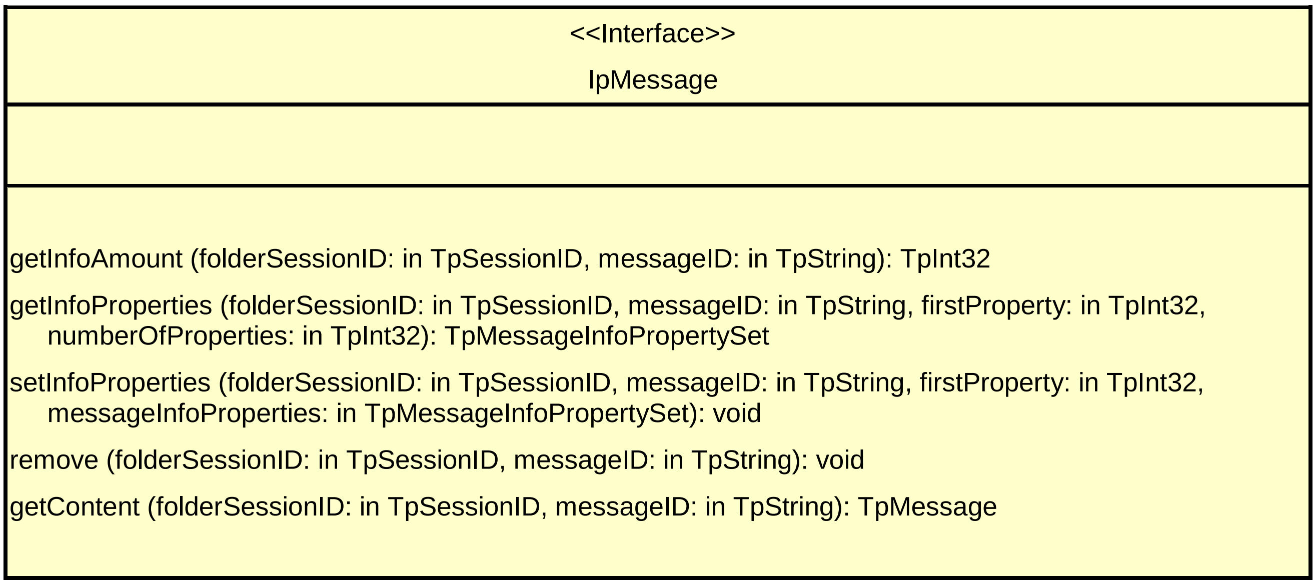
Specifies the identity of the folder. If the folder ID given is not present, the error code P_GMS_INVALID_FOLDER_ID is returned.

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID, P_GMS_INSUFFICIENT_PRIVILEGE, P_GMS_INVALID_FOLDER_ID, P_GMS_FOLDER_IS_OPEN`

8.5 Interface Class IpMessage

Inherits from: IpService.



<<Interface>>

IpMessage

getInfoAmount (folderSessionID: in TpSessionID, messageID: in TpString): TpInt32

getInfoProperties (folderSessionID: in TpSessionID, messageID: in TpString, firstProperty: in TpInt32, numberOfProperties: in TpInt32): TpMessageInfoPropertySet

setInfoProperties (folderSessionID: in TpSessionID, messageID: in TpString, firstProperty: in TpInt32, messageInfoProperties: in TpMessageInfoPropertySet): void

remove (folderSessionID: in TpSessionID, messageID: in TpString): void

getContent (folderSessionID: in TpSessionID, messageID: in TpString): TpMessage

Method
getInfoAmount()

This method returns the number of message information properties of the specified message.

Returns: numberOfProperties

The number of properties associated with the message. The application can then use the information contained to decide whether to get the message or the message information properties from a mailbox folder. The number of properties is zero or positive.

Parameters

folderSessionID: in TpSessionID

This is the session ID of the open folder. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

messageID: in TpString

Specifies the identity of the message. If the message ID given is not present, the error code P_GMS_INVALID_MESSAGE_ID is returned.

Returns

TpInt32

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_GMS_INVALID_MESSAGE_ID

Method
`getInfoProperties()`

This method returns the properties of a message.

Returns: messageInfoProperties

The message information properties (names and values) present in the message. Message properties include message format, read/unread, sent/unsent, message size, relevant dates and times, subject and addresses.

Parameters

`folderSessionID: in TpSessionID`

This is the session ID of the open folder. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

`messageID: in TpString`

Specifies the identity of the message. If the message ID given is not present, the error code P_GMS_INVALID_MESSAGE_ID is returned.

`firstProperty: in TpInt32`

This is the first property of interest. This number represents the starting point where the first property of the list to be retrieved from the message is located. Properties are numbered from zero.

`numberOfProperties: in TpInt32`

The number of properties to return. If the value of this parameter is zero, then all properties will be returned. Otherwise, the value must be a positive number. If the number is not positive, the error code P_GMS_NUMBER_NOT_POSITIVE is returned.

Returns

`TpMessageInfoPropertySet`

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID, P_GMS_NUMBER_NOT_POSITIVE, P_GMS_INVALID_MESSAGE_ID`

Method
`setInfoProperties()`

This method sets the properties of a message.

Parameters

`folderSessionID: in TpSessionID`

This is the session ID of the open folder. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

`messageID: in TpString`

Specifies the identity of the message. If the message ID given is not present, the error code P_GMS_INVALID_MESSAGE_ID is returned.

`firstProperty: in TpInt32`

This is the first property of interest. This number represents the starting point where the first property of the list to be retrieved from the message is located. Properties are numbered from zero.

`messageInfoProperties: in TpMessageInfoPropertySet`

This specifies the message information properties (names and values) to be set in the message. Message properties that may be changed include read/unread status, subject and importance. If the properties cannot be changed, then the error code P_GMS_PROPERTY_NOT_SET is returned.

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID, P_GMS_INVALID_MESSAGE_ID, P_GMS_PROPERTY_NOT_SET`

Method
`remove()`

This method removes a message from the open mailbox folder. If the application does not have sufficient privilege to remove the message, the error code P_GMS_INSUFFICIENT_PRIVILEGE is returned.

Parameters

`folderSessionID: in TpSessionID`

This is the session ID of the open folder. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

`messageID: in TpString`

Specifies the identity of the message. If the message ID given is not present, the error code P_GMS_INVALID_MESSAGE_ID is returned.

The message ID can be obtained by calling the getFolderInfo and getFolderInfoProperties or embedded in an event notification from the messaging service, with information on the mailbox and notifications contained in that operation. If the message cannot be removed, the error code P_GMS_MESSAGE_NOT_REMOVED is returned.

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID, P_GMS_INSUFFICIENT_PRIVILEGE, P_GMS_MESSAGE_NOT_REMOVED, P_GMS_INVALID_MESSAGE_ID`

Method
`getContent()`

This method retrieves the message content.

Returns: content

Returns the message content.

Parameters

`folderSessionID: in TpSessionID`

This is the session ID of the open folder. If the session ID is not a valid session ID, the error code P_INVALID_SESSION_ID is returned.

`messageID: in TpString`

Specifies the identity of the message. If the message ID given is not present, the error code P_GMS_INVALID_MESSAGE_ID is returned.

Returns

`TpMessage`

Raises

`TpCommonExceptions, P_INVALID_SESSION_ID, P_GMS_INVALID_MESSAGE_ID`

9 State Transition Diagrams

There are no State Transition Diagrams for the Generic Messaging SCF.

10 Data Definitions

This clause provides the generic messaging service data definitions necessary to support the API specification. All data types referenced but not defined in this clause are common data definitions which may be found in ES 201 915-2.

10.1 Event notification Definitions

10.1.1 TpMessagingEventName

Defines the name of event being notified. In phase 2 of the APIs, only the following events are supported.

Name	Value	Description
`P_EVENT_GMS_NAME_UNDEFINED`	0	Undefined
`P_EVENT_GMS_NEW_MESSAGE_ARRIVED`	1	GMS -New Message Arrived

10.1.2 TpMessagingEventCriteria

Defines the `Tagged Choice of Data Elements` that specify the criteria for an event notification to be generated.

	Tag Element Type	
	TpMessagingEventName	

Tag Element Value	Choice Element Type	Choice Element Name
P_EVENT_GMS_NEW_MESSAGE_ARRIVED	TpGMSNewMessageArrivedCriteria	`EventGMSNewMessageArrived`

10.1.3 TpGMSNewMessageArrivedCriteria

Defines the `Sequence of Data Elements` that specify the criteria for a GMS New Message Arrived event.

Sequence Element Name	Sequence Element Type
`MailboxID`	`TpAddress`
`AuthenticationInfo`	`TpString`

10.1.4 TpMessagingEventInfo

Defines the Tagged Choice of Data Elements that specify the information returned to the application in an event notification.

	Tag Element Type	
	TpMessagingEventName	

Tag Element Value	Choice Element Type	Choice Element Name
P_EVENT_GMS_NAME_UNDEFINED	TpString	EventNameUndefined
P_EVENT_GMS_NEW_MESSAGE_ARRIVED	TpGMSNewMessageArrivedInfo	EventGMSNewMessage Arrived

10.1.5 TpGMSNewMessageArrivedInfo

Defines the Sequence of Data Elements that specify the information returned to the application in a GMS New Message Arrived event.

Sequence Element Name	Sequence Element Type
MailboxID	TpAddress
FolderID	TpString
MessageID	TpString
NumberOfProperties	TpInt32

10.2 Generic Messaging Data Definitions

10.2.1 IpMessagingManager

Defines the address of an IpMessagingManager Interface.

10.2.2 IpMessagingManagerRef

Defines a Reference to type IpMessagingManager.

10.2.3 IpAppMessagingManager

Defines the address of an IpAppMessagingManager Interface.

10.2.4 IpAppMessagingManagerRef

Defines a Reference to type IpAppMessagingManager.

10.2.5 IpMailbox

Defines the address of an IpMailbox Interface.

10.2.6 IpMailboxRef

Defines a Reference to type IpMailbox.

10.2.7 IpMailboxFolder

Defines the address of an `IpMailboxFolder` Interface.

10.2.8 IpMailboxFolderRef

Defines a `Reference` to type IpMailboxFolder.

10.2.9 IpMessage

Defines the address of an `IpMessage` Interface.

10.2.10 IpMessageRef

Defines a `Reference` to type IpMessage.

10.2.11 TpFolderInfoProperty

Defines the `Tagged Choice of Data Elements` that specify the information properties of a folder.

	Tag Element Type	
	TpFolderInfoPropertyName	

Tag Element Value	Choice Element Type	Choice Element Name
P_MESSAGING_FOLDER_ID	TpString	MessagingFolderID
P_MESSAGING_FOLDER_MESSAGE	TpString	MessagingFolderMessage
P_MESSAGING_FOLDER_SUBFOLDER	TpString	MessagingFolderSubfolder
P_MESSAGING_FOLDER_DATE_CREATED	TpDateAndTime	MessagingFolderDateCreated
P_MESSAGING_FOLDER_DATE_CHANGED	TpDateAndTime	MessagingFolderDateChanged

10.2.12 TpFolderInfoPropertyName

Defines a specific folder information property name.

Name	Value	Description
P_MESSAGING_FOLDER_UNDEFINED	0	Undefined
P_MESSAGING_FOLDER_ID	1	The fully qualified ID of this folder (i.e. including parent folder ID and mailbox ID)
P_MESSAGING_FOLDER_MESSAGE	2	Indicates the ID of a message
P_MESSAGING_FOLDER_SUBFOLDER	3	The fully qualified ID of a subfolder (i.e. including parent folder ID and mailbox ID)
P_MESSAGING_FOLDER_DATE_CREATED	4	Indicates the date created
P_MESSAGING_FOLDER_DATE_CHANGED	5	Indicates the date last changed

10.2.13 TpFolderInfoPropertySet

Defines a `Numbered Set of Data Elements` of TpFolderInfoProperty.

10.2.14 TpMailboxFolderIdentifier

Defines the Sequence of Data Elements that identify a folder.

Sequence Element Name	Sequence Element Type
Mailbox folder	IpMailboxFolderRef
SessionID	TpSessionID

10.2.15 TpMailboxIdentifier

Defines the Sequence of Data Elements that identify a mailbox.

Sequence Element Name	Sequence Element Type
Mailbox	IpMailboxRef
SessionID	TpSessionID

10.2.16 TpMailboxInfoProperty

Defines the Tagged Choice of Data Elements that specify the information properties of a mailbox.

	Tag Element Type	
	TpMailboxInfoPropertyName	

Tag Element Value	Choice Element Type	Choice Element Name
P_MESSAGING_MAILBOX_ID	TpAddress	MessagingMailboxID
P_MESSAGING_MAILBOX_OWNER	TpString	MessagingMailboxOwner
P_MESSAGING_MAILBOX_FOLDER	TpString	MessagingMailboxFolder
P_MESSAGING_MAILBOX_DATE_CREATED	TpDateAndTime	MessagingMailboxDateCreated
P_MESSAGING_MAILBOX_DATE_CHANGED	TpDateAndTime	MessagingMailboxDateChanged

10.2.17 TpMailboxInfoPropertyName

Defines a specific mailbox information property name.

Name	Value	Description
P_MESSAGING_MAILBOX_UNDEFINED	0	Undefined
P_MESSAGING_MAILBOX_ID	1	The ID of the Mailbox
P_MESSAGING_MAILBOX_OWNER	2	The owner of the mailbox
P_MESSAGING_MAILBOX_FOLDER	3	The fully qualified ID of a folder (i.e. including parent folder ID and mailbox ID)
P_MESSAGING_MAILBOX_DATE_CREATED	4	Indicates the date created
P_MESSAGING_MAILBOX_DATE_CHANGED	5	Indicates the date last changed

10.2.18 TpMailboxInfoPropertySet

Defines a Numbered Set of Data Elements of TpMailboxInfoProperty.

10.2.19 TpMessage

This data type is identical to a TpLongstring, and defines the message content.

10.2.20 TpMessageFormat

Defines the format of a message.

Name	Value	Description
P_MESSAGING_MESSAGE_FORMAT_UNDEFINED	0	Undefined
P_MESSAGING_MESSAGE_FORMAT_TEXT	1	Non-specific text format
P_MESSAGING_MESSAGE_FORMAT_BINARY	2	Non-specific binary format
P_MESSAGING_MESSAGE_FORMAT_UUENCODED	3	UUENCODED format
P_MESSAGING_MESSAGE_FORMAT_MIME	4	MIME format
P_MESSAGING_MESSAGE_FORMAT_WAVE	5	WAVE audio format
P_MESSAGING_MESSAGE_FORMAT_AU	6	AU audio format

10.2.21 TpMessageInfoProperty

Defines the Tagged Choice of Data Elements that specify the information properties of a message.

	Tag Element Type	
	TpMessageInfoPropertyName	

Tag Element Value	Choice Element Type	Choice Element Name
P_MESSAGING_MESSAGE_ID	TpString	MessagingMessageID
P_MESSAGING_MESSAGE_SUBJECT	TpString	MessagingMessageSubject
P_MESSAGING_MESSAGE_DATE_SENT	TpDateAndTime	MessagingMessageDateSent
P_MESSAGING_MESSAGE_DATE_RECEIVED	TpDateAndTime	MessagingMessageDate Received
P_MESSAGING_MESSAGE_DATE_CHANGED	TpDateAndTime	MessagingMessageDateChanged
P_MESSAGING_MESSAGE_SENT_FROM	TpAddress	MessagingMessageSentFrom
P_MESSAGING_MESSAGE_SENT_TO	TpAddress	MessagingMessageSentTo
P_MESSAGING_MESSAGE_CC_TO	TpAddress	MessagingMessageCCTo
P_MESSAGING_MESSAGE_BCC_TO	TpAddress	MessagingMessageBCCTo
P_MESSAGING_MESSAGE_SIZE	TpInt32	MessagingMessageSize
P_MESSAGING_MESSAGE_PRIORITY	TpMessagePriority	MessagingMessagePriority
P_MESSAGING_MESSAGE_FORMAT	TpMessageFormat	MessagingMessageFormat
P_MESSAGING_MESSAGE_FOLDER	TpString	MessagingMessageFolder
P_MESSAGING_MESSAGE_STATUS	TpMessageStatus	MessagingMessageStatus

10.2.22 TpMessageInfoPropertyName

Defines a specific message information property name.

Name	Value	Description
P_MESSAGING_MESSAGE_UNDEFINED	0	Undefined
P_MESSAGING_MESSAGE_ID	1	The identity of the message
P_MESSAGING_MESSAGE_SUBJECT	2	The subject of the message
P_MESSAGING_MESSAGE_DATE_SENT	3	Indicates the date send
P_MESSAGING_MESSAGE_DATE_RECEIVED	4	Indicates the date received
P_MESSAGING_MESSAGE_DATE_CHANGED	5	Indicates the date last changed
P_MESSAGING_MESSAGE_SENT_FROM	6	Indicates the sender
P_MESSAGING_MESSAGE_SENT_TO	7	Indicates the Sent To addressees
P_MESSAGING_MESSAGE_CC_TO	8	Indicates the Copied To addressees
P_MESSAGING_MESSAGE_BCC_TO	9	Indicates the Copied Blind addressees
P_MESSAGING_MESSAGE_SIZE	10	Indicates the size of the message in bytes
P_MESSAGING_MESSAGE_PRIORITY	11	Indicates the priority of the message
P_MESSAGING_MESSAGE_FORMAT	12	Indicates the format of the message
P_MESSAGING_MESSAGE_FOLDER	13	The fully qualified ID of the folder in which the message is stored
P_MESSAGING_MESSAGE_STATUS	14	The status of the message

10.2.23 TpMessageInfoPropertySet

Defines a `Numbered Set of Data Elements` of TpMessageInfoProperty.

10.2.24 TpMessagePriority

Defines the priority of a message.

Name	Value	Description
P_MESSAGING_MESSAGE_PRIORITY_UNDEFINED	0	Undefined/Normal
P_MESSAGING_MESSAGE_PRIORITY_HIGH	1	High priority
P_MESSAGING_MESSAGE_PRIORITY_LOW	2	Low priority

10.2.25 TpMessageStatus

Defines the status of a message.

Name	Value	Description
P_MESSAGING_MESSAGE_STATUS_READ_MESSAGE	0	Read message
P_MESSAGING_MESSAGE_STATUS_UNREAD_MESSAGE	1	Unread message
P_MESSAGING_MESSAGE_STATUS_FORWARDED_MESSAGE	2	Forwarded message
P_MESSAGING_MESSAGE_STATUS_REPLIED_TO_MESSAGE	3	Replied to message
P_MESSAGING_MESSAGE_STATUS_SAVED_OR_UNSENT_MESSAGE	4	Saved or unsent message
P_MESSAGING_MESSAGE_STATUS_NOTIFICATION_THAT_A_MESSAGE_WAS_DELIVERED	5	Notification of a delivered message
P_MESSAGING_MESSAGE_STATUS_NOTIFICATION_THAT_A_MESSAGE_WAS_READ	6	Notification of a read message
P_MESSAGING_MESSAGE_STATUS_NOTIFICATION_THAT_A_MESSAGE_WAS_NOT_DELIVERED	7	Notification of a message that was not delivered
P_MESSAGING_MESSAGE_STATUS_NOTIFICATION_THAT_A_MESSAGE_WAS_NOT_READ	8	Notification of a message that was not read

10.2.26 TpMessagingFault

Defines the cause of the messaging fault detected.

Name	Value	Description
P_MESSAGING_FAULT_UNDEFINED	0	Undefined

11 Exception Classes

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

Name	Description
P_GMS_CANNOT_UNLOCK_MAILBOX	Mailbox is locked by another application
P_GMS_FOLDER_IS_OPEN	Action cannot be performed because the folder is open
P_GMS_INSUFFICIENT_PRIVILEGE	The application has insufficient privilege to perform this action
P_GMS_INVALID_AUTHENTICATION_INFORMATION	Authentication Information is not valid
P_GMS_INVALID_FOLDER_ID	The folder ID is invalid (does not exist if opening a folder, already exists if creating a folder)
P_GMS_INVALID_MAILBOX	Chosen Mailbox Address is invalid
P_GMS_INVALID_MESSAGE_ID	The message ID is invalid / does not exist
P_GMS_LOCKING_LOCKED_MAILBOX	Attempt to lock an already locked mailbox
P_GMS_MAILBOX_LOCKED	Action cannot be performed because the mailbox is locked by another application
P_GMS_MAILBOX_OPEN	Action cannot be performed because the mailbox is already open
P_GMS_MESSAGE_NOT_REMOVED	The message cannot be removed
P_GMS_NUMBER_NOT_POSITIVE	A negative number of properties was requested
P_GMS_PROPERTY_NOT_SET	The property cannot be changed
P_GMS_UNLOCKING_UNLOCKED_MAILBOX	Attempt to unlock an unlocked mailbox

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 (normative): OMG IDL Description of Generic Messaging SCF

The OMG IDL representation of this interface specification is contained in a text file (gms.idl contained in archive es_20191509v010201m0.ZIP) which accompanies the present document.

Annex B (informative): Summary of differences between V1.1.1 (Parlay 3.0) and V1.2.1 (Parlay 3.1)

B.1 IpService

setCallback() and setCallbackWithSessionID() now both raise P_INVALID_INTERFACE_TYPE.

B.2 IpMailboxFolder

putMessage (folderSessionID: in TpSessionID, message: in ~~IpMessageRef~~TpMessage, messageInfoProperties: in TpMessageInfoPropertySet): void

getMessage (folderSessionID: in TpSessionID, messageID: in TpString~~, removeMessage: in TpBoolean~~): IpMessageRef

B.3 IpMessage

Method getContent() added.

getContent (folderSessionID: in TpSessionID, messageID: in TpString): TpMessage

B.4 Data Types

TpMailboxFolderIdentifier

Defines `the Sequence of Data Elements` that identify a folder.

Sequence Element Name	Sequence Element Type
`Mailbox`~~ f~~`Folder`	`IpMailboxFolderRef`
`SessionID`	`TpSessionID`

History

Document history		
V1.1.1	February 2002	Publication
V1.2.1	May 2002	Membership Approval Procedure MV 20020705: 2002-05-07 to 2002-07-05