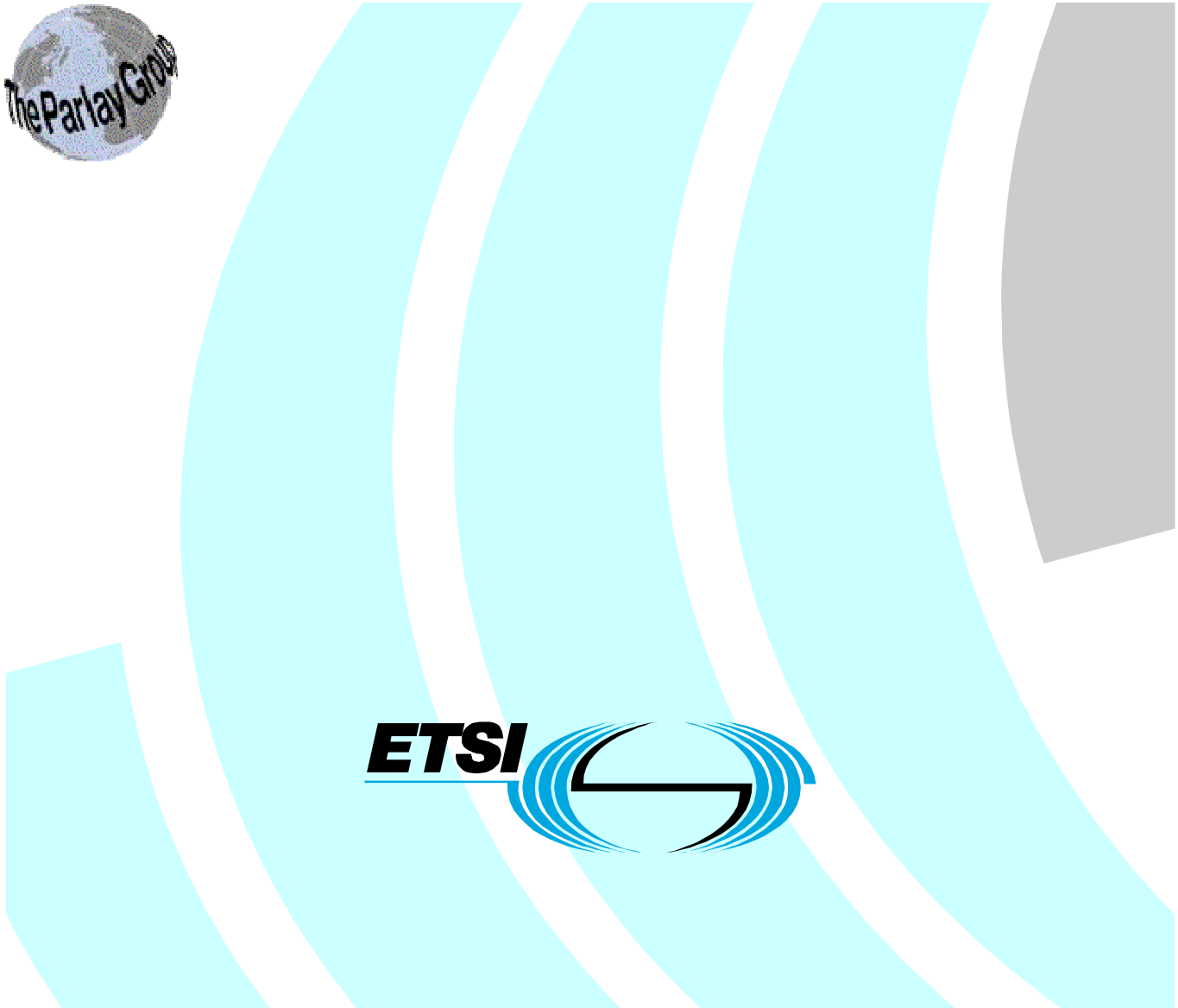
ETSI ES 203 915-11 V1.1.1 (2005-04)

ETSI Standard

Open Service Access (OSA); Application Programming Interface (API); Part 11: Account Management SCF (Parlay 5)



Reference
DES/TISPAN-01005-11-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/chaircor/ETSI_support.asp

Copyright Notification

Contents

Intellectual Property Rights

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

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

Foreword

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

The present document is part 11 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.

The present document is equivalent to 3GPP TS 29.198-11 V6.2.0 (Release 6).

1 Scope

The present document is part 11 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 Account Management Service Capability Feature (SCF) aspects of the interface. All aspects of the Account 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.
- WSDL 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 Account Management SCF

The following clauses describe each aspect of the Account Management 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 shows 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 shows 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 ES 203 915-2.

4.1 General requirements on support of methods

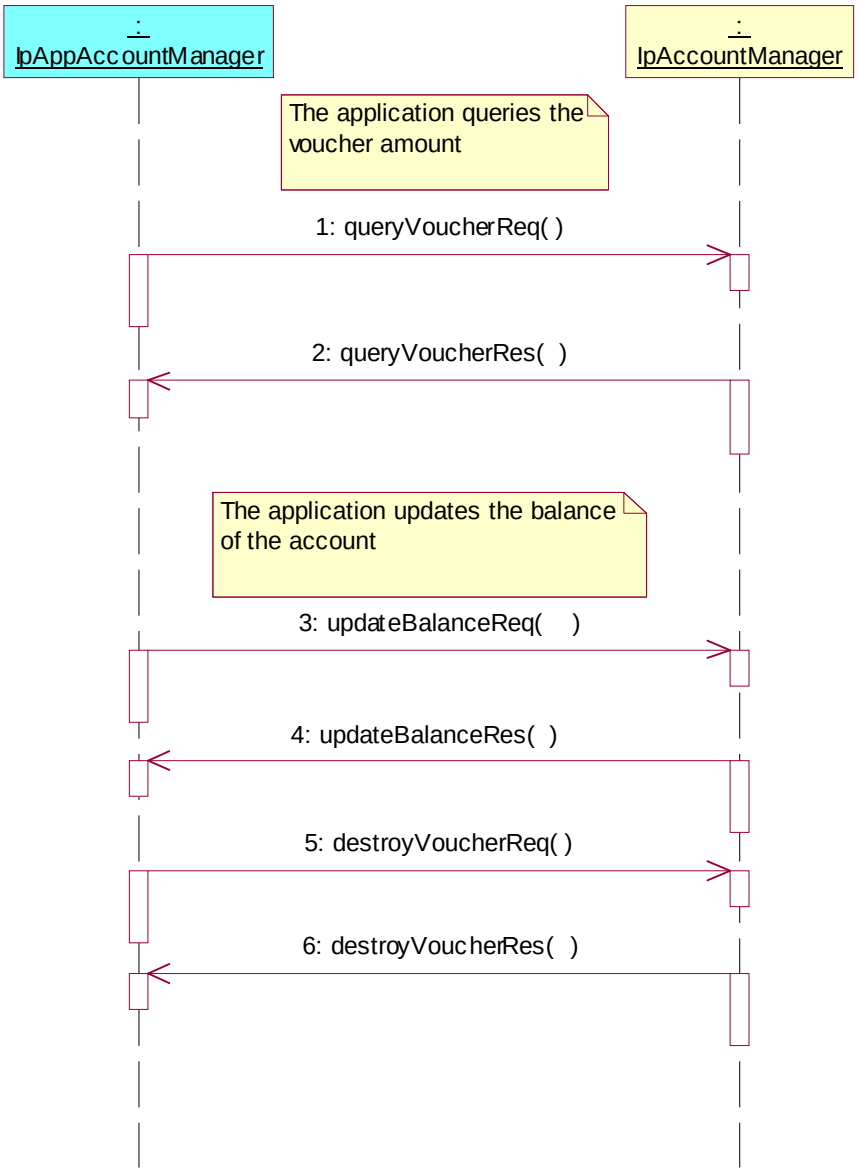
An implementation of this API which supports or implements a method described in the present document, shall support or implement the functionality described for that method, for at least one valid set of values for the parameters of that method.

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.

Where a method is not supported by an implementation of an Application interface, a call to that method shall be possible, and no exception shall be returned.

5 Sequence Diagrams

5.1 Standard Voucher Handling



1: This message is used to get a voucher for an amount.

2: This message used to return the amount requested.

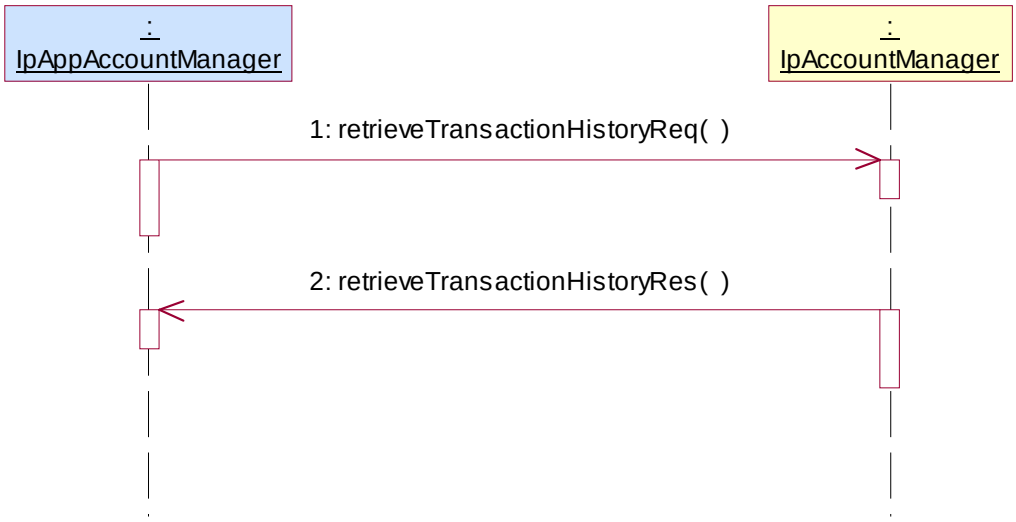
3: This message is used to update the balance of the account.

4: This message is used to return the confirmation of the update.

5: This message is used to remove the voucher.

6: This message is used to confirm that the voucher is removed.

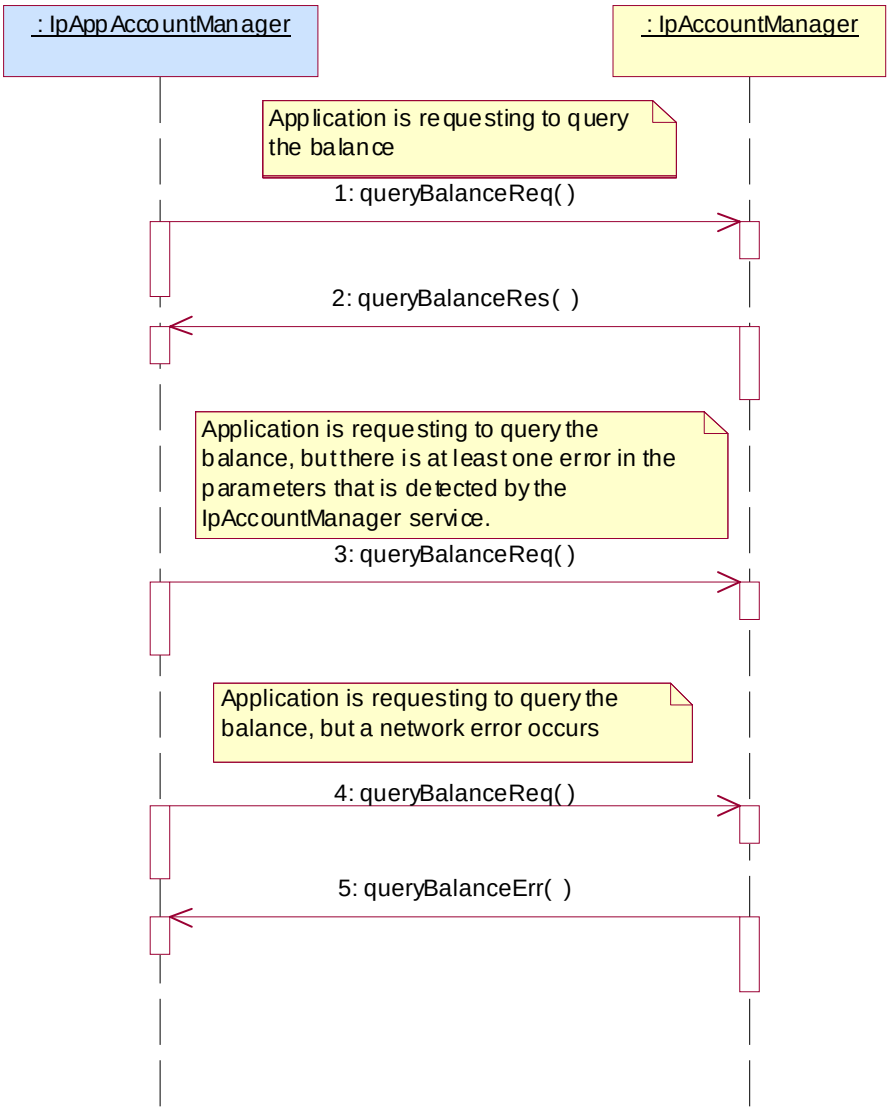
5.2 Standard Transaction History Retrieval



1: This message is used by the application to retrieve a transaction history for a certain subscriber's account.

2: This method passes the result of the transaction history retrieval request for a specific user to its callback object.

5.3 Standard Query Handling



1: This message is used to query the balance of the account of one or several users.

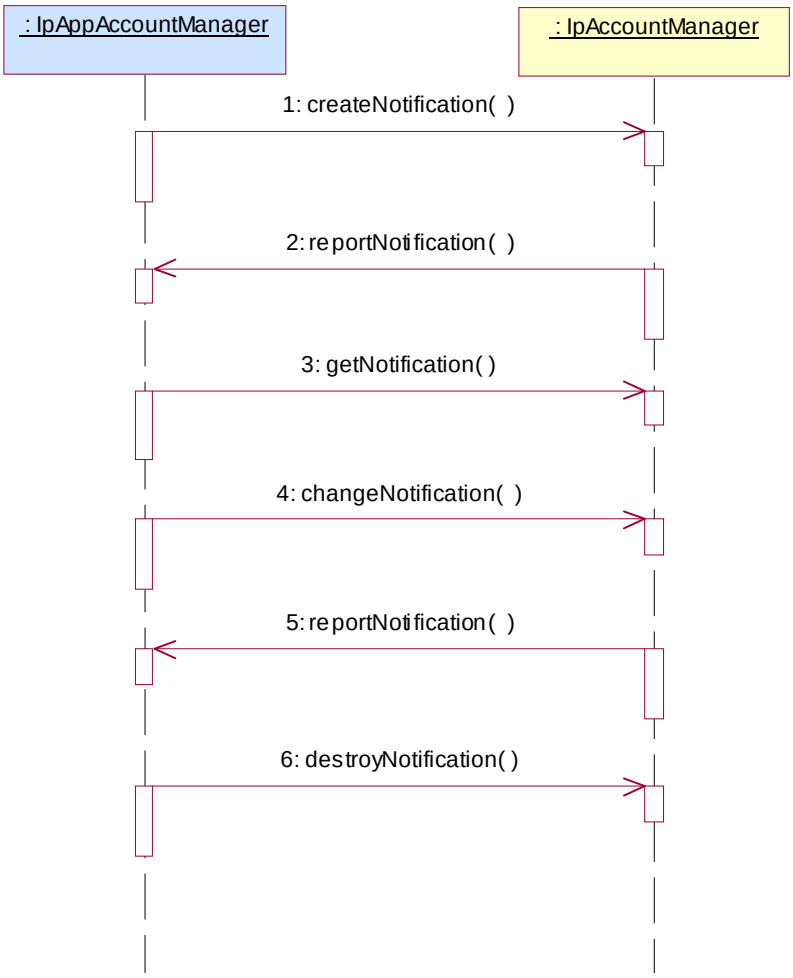
2: This message passes the result of the balance query for one or several users to its callback object.

3: This scenario shows the case where at least one error in the parameters of the message is detected by the IpAccountManager object. An exception will be thrown.

4: This scenario shows the case where a network error occurs.

5: This message passes the error of the balance query. No exception is thrown.

5.4 Standard Notification handling



1: This message is used by the application to request notifications from the IpAccountManager service on certain criteria for one or several users.

2: This message is used by the IpAccountManager service to report a charging event that meets the criteria set in the createNotification message.

3: The application can request the current criteria set in the IpAccountManager service by invoking the getNotification method.

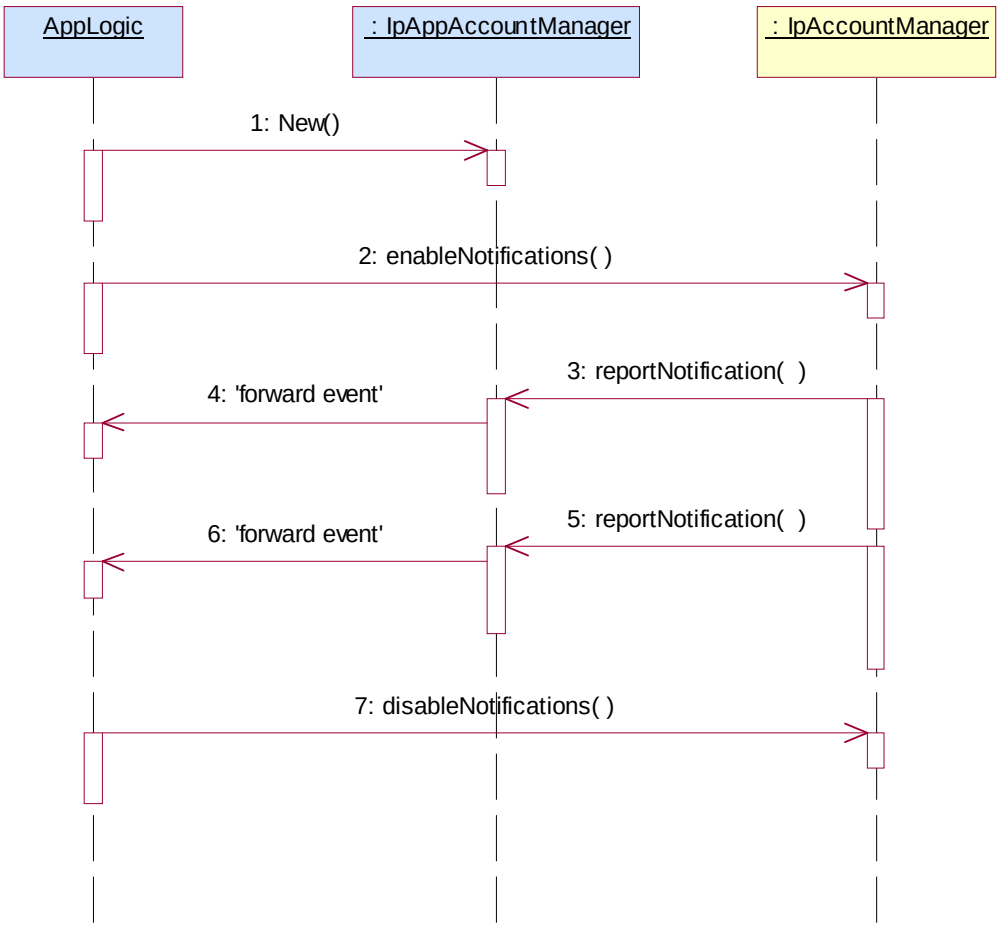
4: This message is used by the application to change the criteria initially created by createNotification, and previously obtained by getNotification.

5: This message is used by the IpAccountManager service to report a charging event that meets the new criteria.

6: This method is used by the application to disable the charging notifications.

5.5 Network Controlled Notifications

The following sequence diagram shows how an application can receive notifications that have not been created by the application, but are provisioned from within the network.



1: The application is started. The application creates a new IpAppAccountManager to handle callbacks.

2: The enableNotifications method is invoked on the IpAccountManager interface to indicate that the application is ready to receive notifications that are created in the network. For illustrative purposes we assume notifications of type "B" are created in the network.

3: When a network created trigger occurs the application is notified on the callback interface.

4: The event is forwarded to the application.

5: When a network created trigger occurs the application is notified on the callback interface.

6: The event is forwarded to the application.

7: When the application does not want to receive notifications created in the network anymore, it invokes disableNotifications on the IpMultiPartyCallConrolManager interface. From now on the gateway will not send any notifications to the application that are created in the network. The application will still receive notifications that it has created himself until the application removes them.

6 Class Diagrams

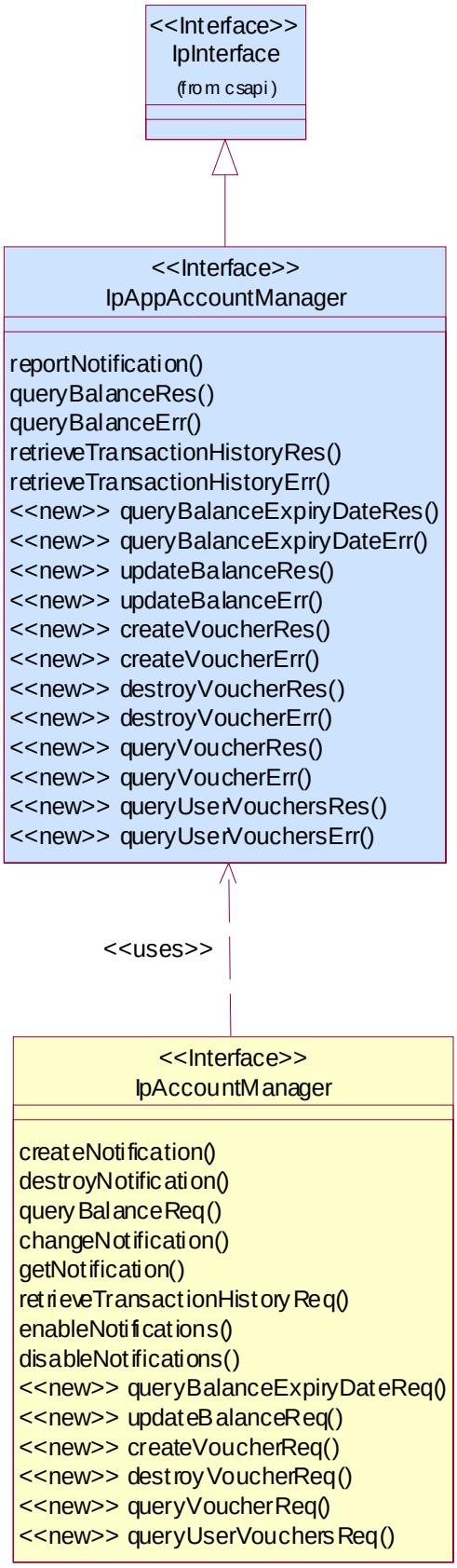


Figure 1: Application Interfaces

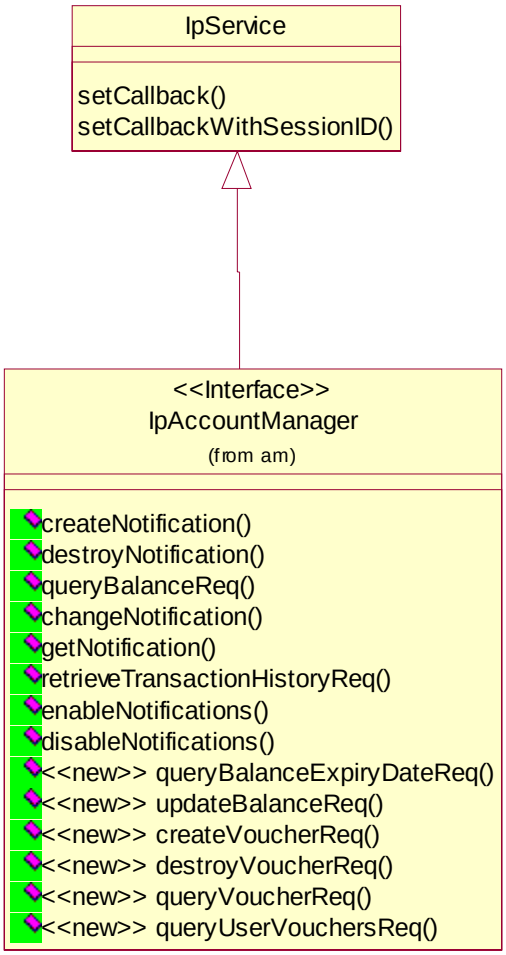


Figure 2: Service Interfaces

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 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.

<<Interface>> IpInterface

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

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`

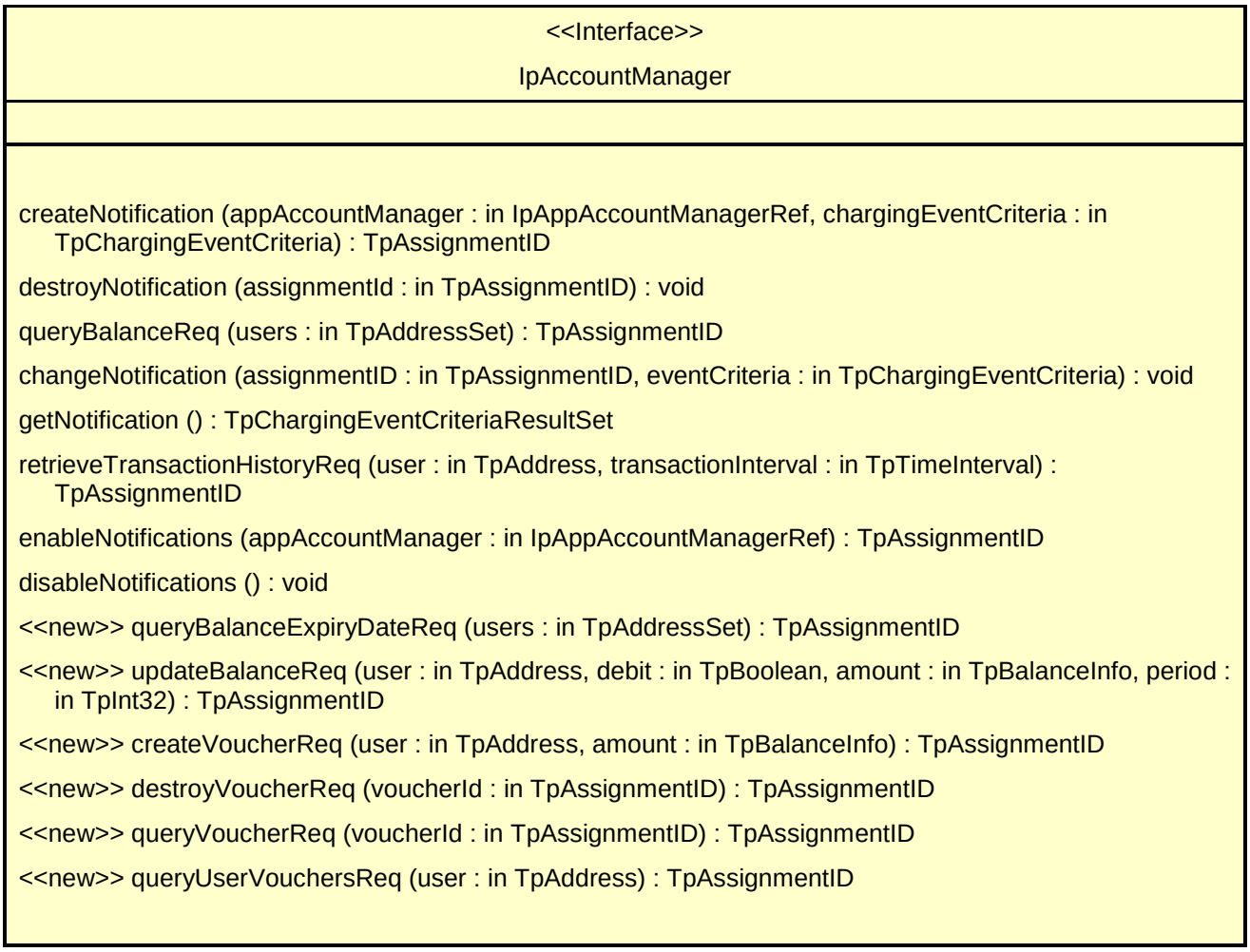
8 Account Management Interface Classes

8.1 Interface Class IpAccountManager

Inherits from: IpService.

The account manager interface provides methods for managing accounts. Applications can use this interface to enable or disable charging-related event notifications and to manage account balances. Vouchers allow indirect references to amounts that can be applied to the account.

This interface shall be implemented by an Account Management SCF. The queryBalanceReq() method, or the retrieveTransactionHistoryReq() method, or both the createNotification() and destroyNotification methods, or both the enableNotifications and disableNotifications methods shall be implemented as a minimum requirement.



8.1.1 Method createNotification()

This method is used by the application to enable charging event notifications to be sent to the application.

If the same application invokes this method multiple times with exactly the same criteria but with different callback references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. 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.

In case the createNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallback().

Returns assignmentId : Specifies the ID assigned by the account management object for this newly enabled event notification.

Parameters

`appAccountManager : in IpAppAccountManagerRef`

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.

`chargingEventCriteria : in TpChargingEventCriteria`

Specifies the event specific criteria used by the application to define the charging event required. Individual addresses or address ranges may be specified for subscriber accounts. Example of events are "charging" and "recharging".

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_ADDRESS, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE, P_UNKNOWN_SUBSCRIBER

8.1.2 Method destroyNotification()

This method is used by the application to disable charging notifications. This method only applies to notifications created with createNotification().

Parameters

assignmentId : in TpAssignmentID

Specifies the assignment ID that was given by the account management object when the application enabled the charging notification.

Raises

TpCommonExceptions, P_INVALID_ASSIGNMENT_ID

8.1.3 Method queryBalanceReq()

This method is used by the application to query the balance of an account for one or several users. A queryBalanceRes() will be sent for all valid users, and a queryBalanceErr() will be sent for all invalid users or other errors.

Returns queryId : Specifies the ID of the balance query request.

Parameters

users : in TpAddressSet

Specifies the user(s) for which the balance is queried.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_UNKNOWN_SUBSCRIBER, P_UNAUTHORIZED_APPLICATION

8.1.4 Method changeNotification()

This method is used by the application to change the event criteria introduced with createNotification. Any stored criteria associated with the specified assignmentID will be replaced with the specified criteria.

Parameters

assignmentID : in TpAssignmentID

Specifies the ID assigned by the manager interface for the event notification.

eventCriteria : in TpChargingEventCriteria

Specifies the new set of event criteria used by the application to define the event required. Only events that meet these criteria are reported.

Raises

TpCommonExceptions, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE, P_UNKNOWN_SUBSCRIBER, P_INVALID_ADDRESS

8.1.5 Method getNotification()

This method is used by the application to query the event criteria set with createNotification or changeNotification.

Returns eventCriteria : Specifies the event criteria used by the application to define the event required. Only events that meet these criteria are reported.

Parameters
No Parameters were identified for this method.

Returns

TpChargingEventCriteriaResultSet

Raises

TpCommonExceptions

8.1.6 Method retrieveTransactionHistoryReq()

This asynchronous method is used by the application to retrieve a transaction history of a subscriber's account. The history is a set of Detailed Records.

Returns retrievalID : Specifies the retrieval ID of the transaction history retrieval request.

Parameters

user : in TpAddress

Specifies the subscriber for whose account the transaction history is to be retrieved.

transactionInterval : in TpTimeInterval

Specifies the time interval for which the application history is to be retrieved.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_UNKNOWN_SUBSCRIBER, P_UNAUTHORIZED_APPLICATION, P_INVALID_TIME_AND_DATE_FORMAT

8.1.7 Method enableNotifications()

This method is used to indicate that the application is able to receive which are provisioned from within the network (i.e. these notifications are NOT set using createNotification() but via, for instance, a network management system). If notifications provisioned for this application are created or changed, the application is unaware of this until the notification is reported.

If the same application invokes this method multiple times with different IpAppAccountManager references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. 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.

When this method is used, it is still possible to use createNotification() for service provider provisioned notifications on the same interface as long as the criteria in the network and provided by createNotification() do not overlap. However, it is NOT recommended to use both mechanisms on the same service manager.

The methods changeNotification(), getNotification(), and destroyNotification() do not apply to notifications provisioned in the network and enabled using enableNotifications(). These only apply to notifications created using createNotification().

Returns assignmentID: Specifies the ID assigned by the manager interface for this operation. This ID is contained in any reportNotification() that relates to notifications provisioned from within the network Repeated calls to enableNotifications() return the same assignment ID.

Parameters

appAccountManager : in IpAppAccountManagerRef

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.

Returns

TpAssignmentID

Raises

TpCommonExceptions

8.1.8 Method disableNotifications()

This method is used to indicate that the application is not able to receive notifications for which the provisioning has been done from within the network. (i.e. these notifications that are NOT set using createNotification() but via, for instance, a network management system). After this method is called, no such notifications are reported anymore.

Parameters

No Parameters were identified for this method.

Raises

TpCommonExceptions

8.1.9 Method <<new>> queryBalanceExpiryDateReq()

This method is used by the application to query the expiration date of the account. The returned date is the date the current balance will expire. Nil is returned if the balance does not expire. A queryBalanceExpiryDateRes() will be sent for all valid users, and a queryBalanceExpiryDateErr() will be sent for all invalid users or other errors.

Returns queryId: Specifies the ID of the balance expiry date query request.

Parameters

users : in TpAddressSet

Specifies the user(s) for which the balance expiry date is queried.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_UNKNOWN_SUBSCRIBER, P_UNAUTHORIZED_APPLICATION

8.1.10 Method <<new>> updateBalanceReq()

This method is used by the application to update the balance of the account.

Returns requestId: Specifies the ID of the update balance request.

Parameters

user : in TpAddress

Specifies the user for which the balance update will be done.

debit : in TpBoolean

Specifies that the amount will be subtracted from the balance if the debit parameter is true, or that the amount will be added to the account if the debit parameter is false.

amount : in TpBalanceInfo

Specifies the amount that will be added or subtracted to the user's account. The charge is specified as a currency amount.

period : in TpInt32

Specifies the period at which the balance will expire. The balance is requested to expire in a number of days indicated by the period parameter. The operator's policy may overrule this parameter. If the period parameter is 0, the operator's policy on the balance expiration is always in effect.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_UNKNOWN_SUBSCRIBER, P_UNAUTHORIZED_APPLICATION

8.1.11 Method <<new>> createVoucherReq()

This method is used by the application to create a voucher for an amount.

Returns requestId: Specifies the ID of the voucher create request.

Parameters

user : in TpAddress

Specifies the user for which the voucher create will be done.

amount : in TpBalanceInfo

Specifies the amount of the voucher. The charge is specified as a currency amount.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_UNKNOWN_SUBSCRIBER, P_UNAUTHORIZED_APPLICATION

8.1.12 Method <<new>> destroyVoucherReq()

This method is used by the application to destroy a voucher for an amount.

Returns requestId: Specifies the ID of the destroy voucher request.

Parameters

voucherId : in TpAssignmentID

Specifies the voucher to be destroyed.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_UNAUTHORIZED_APPLICATION, P_INVALID_ASSIGNMENT_ID

8.1.13 Method <<new>> queryVoucherReq()

This method is used by the application to get the voucher information.

Returns queryId: Specifies the identifier for this request.

Parameters

voucherId : in TpAssignmentID

Specifies the voucher to be queried.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_UNAUTHORIZED_APPLICATION, P_INVALID_ASSIGNMENT_ID

8.1.14 Method <<new>> queryUserVouchersReq()

This method is used by the application to get the vouchers for a user.

Returns queryId: Specifies the ID for the get vouchers request.

Parameters

user : in TpAddress

Specifies the user for which the vouchers will be returned.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_UNKNOWN_SUBSCRIBER, P_UNAUTHORIZED_APPLICATION, P_INVALID_ASSIGNMENT_ID

8.2 Interface Class IpAppAccountManager

Inherits from: IpInterface.

The account manager application interface is implemented by the client application developer and is used to handle charging event notifications and query balance responses.

<<Interface>>

IpAppAccountManager

reportNotification (chargingEventInfo : in TpChargingEventInfo, assignmentId : in TpAssignmentID) : void

queryBalanceRes (queryId : in TpAssignmentID, balances : in TpBalanceSet) : void

queryBalanceErr (queryId : in TpAssignmentID, cause : in TpBalanceQueryError) : void

retrieveTransactionHistoryRes (retrievalID : in TpAssignmentID, transactionHistory : in TpTransactionHistorySet) : void

retrieveTransactionHistoryErr (retrievalID : in TpAssignmentID, transactionHistoryError : in TpTransactionHistoryStatus) : void

<<new>> queryBalanceExpiryDateRes (queryId : in TpAssignmentID, dates : in TpBalanceExpiryDateSet) : void

<<new>> queryBalanceExpiryDateErr (queryId : in TpAssignmentID, cause : in TpBalanceQueryError) : void

<<new>> updateBalanceRes (requestId : in TpAssignmentID, balance : in TpBalance) : void

<<new>> updateBalanceErr (requestId : in TpAssignmentID, cause : in TpBalanceQueryError) : void

<<new>> createVoucherRes (requestId : in TpAssignmentID, voucherId : in TpAssignmentID) : void

<<new>> createVoucherErr (requestId : in TpAssignmentID, cause : in TpVoucherError) : void

<<new>> destroyVoucherRes (requestId : in TpAssignmentID, voucherId : in TpAssignmentID) : void

<<new>> destroyVoucherErr (requestId : in TpAssignmentID, voucherId : in TpAssignmentID, cause : in TpVoucherError) : void

<<new>> queryVoucherRes (queryId : in TpAssignmentID, voucher : in TpVoucher) : void

<<new>> queryVoucherErr (queryId : in TpAssignmentID, voucherId : in TpAssignmentID, cause : in TpVoucherError) : void

<<new>> queryUserVouchersRes (queryId : in TpAssignmentID, vouchers : in TpVoucherSet) : void

<<new>> queryUserVouchersErr (queryId : in TpAssignmentID, cause : in TpVoucherError) : void

8.2.1 Method reportNotification()

This method is used to notify the application of a charging event.

Parameters

`chargingEventInfo : in TpChargingEventInfo`

Specifies data associated with this charging event. These data include the charging event being notified, the current value of the balance after the notified event occurred, and the time at which the charging event occurred.

assignmentId : in TpAssignmentID

Specifies the assignment ID that was returned by the createNotification() method. The application can use the assignment ID to associate events with event-specific criteria and to act accordingly.

8.2.2 Method queryBalanceRes()

This method indicates that the request to query the balance was successful and it reports the requested balance of an account to the application.

Parameters

queryId : in TpAssignmentID

Specifies the ID of the balance query request.

balances : in TpBalanceSet

Specifies the balance for one or more user accounts.

8.2.3 Method queryBalanceErr()

This method indicates that the request to query the balance failed and it reports the cause of failure to the application.

Parameters

queryId : in TpAssignmentID

Specifies the ID of the balance query request.

cause : in TpBalanceQueryError

Specifies the error that led to the failure.

8.2.4 Method retrieveTransactionHistoryRes()

This method indicates that the request to retrieve the transaction history was successful and it returns the requested transaction history.

Parameters

retrievalID : in TpAssignmentID

Specifies the retrievalID of the transaction history retrieval request.

transactionHistory : in TpTransactionHistorySet

Specifies the requested transaction history.

8.2.5 Method retrieveTransactionHistoryErr()

This method indicates that the request to retrieve the transaction history failed and it reports the cause of failure to the application.

Parameters

retrievalID : in TpAssignmentID

Specifies the retrievalID of the transaction history retrieval request.

transactionHistoryError : in TpTransactionHistoryStatus

Specifies the error that occurred while retrieving the transaction history.

8.2.6 Method <<new>> queryBalanceExpiryDateRes()

This method indicates that the request to query the balance expiry date was successful and it reports the requested balance expiry date of an account to the application.

Parameters

`queryId : in TpAssignmentID`

Specifies the ID of the balance query expiry date request.

`dates : in TpBalanceExpiryDateSet`

Specifies the balance expiry date for one or more user accounts.

8.2.7 Method <<new>> queryBalanceExpiryDateErr()

This method indicates that the request to query the balance expiry date failed and it reports the cause of failure to the application.

Parameters

`queryId : in TpAssignmentID`

Specifies the ID of the balance query expiry date request.

`cause : in TpBalanceQueryError`

Specifies the error that led to the failure.

8.2.8 Method <<new>> updateBalanceRes()

This method indicates that the request to update the balance update was successful and it reports the balance of an account to the application.

Parameters

`requestId : in TpAssignmentID`

Specifies the ID of the balance update request.

`balance : in TpBalance`

Specifies the balance of the account.

8.2.9 Method <<new>> updateBalanceErr()

This method indicates that the request to update the balance failed and it reports the cause of failure to the application.

Parameters

`requestId : in TpAssignmentID`

Specifies the ID of the balance update request.

`cause : in TpBalanceQueryError`

Specifies the error that led to the failure.

8.2.10 Method <<new>> createVoucherRes()

This method indicates that the request to create a voucher was successful.

Parameters

`requestId : in TpAssignmentID`

Specifies the ID of the voucher create request.

`voucherId : in TpAssignmentID`

Specifies the voucher.

8.2.11 Method <<new>> createVoucherErr()

This method indicates that the request to create a voucher failed and it reports the cause of failure to the application.

Parameters

`requestId : in TpAssignmentID`

Specifies the ID of the voucher create request.

`cause : in TpVoucherError`

Specifies the error that led to the failure.

8.2.12 Method <<new>> destroyVoucherRes()

This method indicates that the request to destroy a voucher was successful.

Parameters

`requestId : in TpAssignmentID`

Specifies the ID of the request.

`voucherId : in TpAssignmentID`

Specifies the voucher.

8.2.13 Method <<new>> destroyVoucherErr()

This method indicates that the request to destroy a voucher failed and it reports the cause of failure to the application.

Parameters

`requestId : in TpAssignmentID`

Specifies the ID of the request.

`voucherId : in TpAssignmentID`

Specifies the voucher.

`cause : in TpVoucherError`

Specifies the error that led to the failure.

8.2.14 Method <<new>> queryVoucherRes()

This method indicates that the request to query the voucher information was successful and it reports the amount to the application.

Parameters

queryId : in TpAssignmentID

Specifies the ID of the query voucher amount request.

voucher : in TpVoucher

Specifies the information of the voucher.

8.2.15 Method <<new>> queryVoucherErr()

This method indicates that the request to query the voucher failed and it reports the cause of failure to the application.

Parameters

queryId : in TpAssignmentID

Specifies the ID of the query voucher amount request.

voucherId : in TpAssignmentID

Specifies the voucher.

cause : in TpVoucherError

Specifies the error that led to the failure.

8.2.16 Method <<new>> queryUserVouchersRes()

This method indicates that the request to query the vouchers was successful and it reports the set of vouchers to the application.

Parameters

queryId : in TpAssignmentID

Specifies the ID of the query vouchers request.

vouchers : in TpVoucherSet

Specifies the set of vouchers for the user.

8.2.17 Method <<new>> queryUserVouchersErr()

This method indicates that the request to query the vouchers for a user failed and it reports the cause of failure to the application.

Parameters

queryId : in TpAssignmentID

Specifies the ID of the query vouchers request.

cause : in TpVoucherError

Specifies the error that led to the failure.

9 State Transition Diagrams

9.1 State Transition Diagrams for IpAccountManager

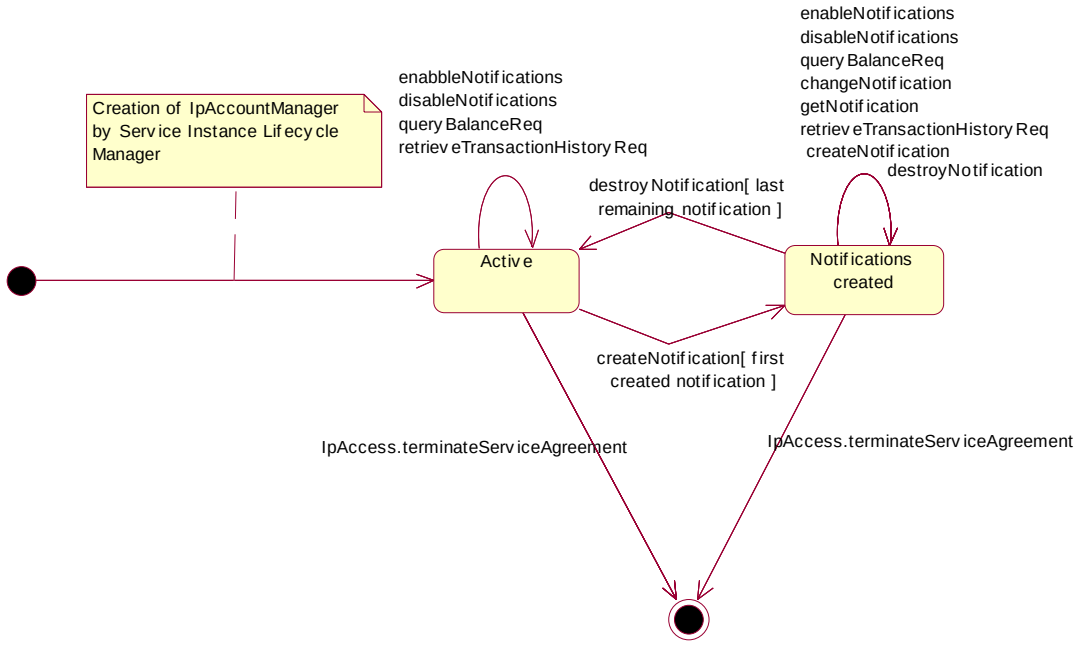


Figure 3: Application view on the IpAccountManager

9.1.1 Active State

In this state a relation between the Application and the Account Management has been established. The state allows the application to indicate that it is interested in charging related events, by calling createNotification/enableNotifications. In case such an event occurs, Account Manager will inform the application by invoking the operation reportNotification() on the IpAppAccountManager interface. The application can also indicate it is no longer interested in certain charging related events by calling destroyNotification/disableNotifications.

9.1.2 Notifications created State

When the Account Manager is in the Notifications created state, events requested with createNotification/enableNotifications will be forwarded to the application. In this state the application can request to change the notifications or query the Account Manager for the notifications currently set.

10 Account Management Service Properties

The following table lists properties relevant for the Account Management API.

Property	Type	Description/Interpretation
P_EVENT_TYPES	INTEGER_SET	Indicates the event types supported by the SCS. Static events are the events by which applications are initiated.
P_ADDRESSPLAN	INTEGER_SET	Indicates the supported address plans (defined in TpAddressPlan.) E.g. {P_ADDRESS_PLAN_E164, P_ADDRESS_PLAN_IP}). Note that more than one address plan may be supported.

The previous table lists properties related to the capabilities of the SCS itself. The following table lists properties that are used in the context of the Service Level Agreement, e.g. to restrict the access of applications to the capabilities of the SCS.

Property	Type	Description/Interpretation
P_NOTIFICATION_ADDRESS_RANGES	XML_ADDRESS_RANGE_SET	Indicates for which numbers notifications may be set. More than one range may be present. For terminating notifications they apply to the terminating number, for originating notifications they apply only to the originating number.
P_CURRENCY_ALLOWED	STRING_SET	Indicates the currencies that can be returned in the queryBalanceRes. The valid values for the string set are according to ISO 4217: 1995. E.g. {"EUR", "NLG"}.
P_HISTORY_ALLOWED	STRING_SET	Indicates the length of the transaction history interval that is allowed to be retrieved by the application. The valid values for the string are according to TpDateAndTime. The string-set will be of format {"lower_start_time", "upper_stop_time"}, e.g. {"1998-12-04 10:30", "1999-12-04 10:30"}.
P_MAX_ADDRESSES_PER_QUERY	INTEGER_SET	Indicates the maximum number of addresses which can be included in a queryBalanceReq.

11 Data Definitions

11.1 Account Management Data Definitions

This clause provides the Account Management specific data definitions necessary to support the OSA interface specification.

The general format of a data definition specification is the following:

- Data type, that shows the name of the data type.
- Description, that describes the data type.
- Tabular specification, that specifies the data types and values of the data type.
- Example, if relevant, shown to illustrate the data type.

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

11.1.1 IpAppAccountManager

Defines the address of an `IpAppAccountManager` Interface.

11.1.2 IpAppAccountManagerRef

Defines a Reference to type `IpAppAccountManager`.

11.1.3 IpAccountManager

Defines the address of an `IpAccountManager` Interface.

11.1.4 IpAccountManagerRef

Defines a Reference to type `IpAccountManager`.

11.1.5 TpBalanceQueryError

Defines an error that is reported by the Charging service capability feature as a result of a balance query request.

Name	Value	Description
P_BALANCE_QUERY_OK	0	No error occurred while processing the request
P_BALANCE_QUERY_ERROR_UNDEFINED	1	General error, unspecified
P_BALANCE_QUERY_UNKNOWN_SUBSCRIBER	2	Subscriber for which balance is queried is unknown
P_BALANCE_QUERY_UNAUTHORIZED_APPLICATION	3	Application is not authorized to query balance
P_BALANCE_QUERY_SYSTEM_FAILURE	4	System failure. The request could not be handled

11.1.6 TpChargingEventName

Defines the charging event for which notifications can be requested by the application.

Name	Value	Description
P_AM_CHARGING	0	End user's account has been charged by an application
P_AM_RECHARGING	1	End user has recharged the account
P_AM_ACCOUNT_LOW	2	Account balance is below the balance threshold
P_AM_ACCOUNT_ZERO	3	Account balance is at zero
P_AM_ACCOUNT_DISABLED	4	Account has been disabled

11.1.7 TpBalanceInfo

Defines the structure of data elements that specifies detailed balance info.

Structured Member Name	Structured Member Type	Description
Currency	TpString	Currency unit according to ISO 4217: 1995.
ValuePartA	TpInt32	This data type is identical to a TpInt32 and specifies the most significant part of the composed value. A currency amount is composed as follows: ((ValuePartA × 2^{32} + ValuePartB) × 0,0001).
ValuePartB	TpInt32	This data type is identical to a TpInt32 and specifies the least significant part of the composed value.
Exponent	TpInt32	Specifies the position of the decimal point in the currency amount made up of the ValuePartA and the ValuePartB, as described above. E.g. an exponent of 4 means a pure integer value, whereas an exponent of 2 means an accuracy of 0,01.
AdditionalInfo	TpString	Descriptive string, containing additional information, which is sent to the application without prior evaluation.

As an example, the currency amount composed of a Currency of EUR, a ValuePartA of 0, a ValuePartB of 10 000, and an exponent of 2 yields a currency amount of € 100,00.

Valid Currencies are:

```
ADP, AED, AFA, ALL, AMD, ANG, AON, AOR, ARS, ATS, AUD, AWG, AZM, BAM,
BBD, BDT, BEF, BGL, BGN, BHD, BIF, BMD, BND, BOB, BOV, BRL, BSD, BTN,
BWP, BYB, BZD, CAD, CDF, CHF, CLF, CLP, CNY, COP, CRC, CUP, CVE, CYP,
CZK, DEM, DJF, DKK, DOP, DZD, ECS, ECV, EEK, EGP, ERN, ESP, ETB, EUR,
FIM, FJD, FKP, FRF, GBP, GEL, GHC, GIP, GMD, GNF, GRD, GTQ, GWP, GYD,
HKD, HNL, HRK, HTG, HUF, IDR, IEP, ILS, INR, IQD, IRR, ISK, ITL, JMD,
```

JOD, JPY, KES, KGS, KHR, KMF, KPW, KRW, KWD, KYD, KZT, LAK, LBP, LKR, LRD, LSL, LTL, LUF, LVL, LYD, MAD, MDL, MGF, MKD, MMK, MNT, MOP, MRO, MTL, MUR, MVR, MWK, MXN, MXV, MYR, MZM, NAD, NGN, NIO, NLG, NOK, NPR, NZD, OMR, PAB, PEN, PGK, PHP, PKR, PLN, PTE, PYG, QAR, ROL, RUB, RUR, RWF, SAR, SBD, SCR, SDD, SEK, SGD, SHP, SIT, SKK, SLL, SOS, SRG, STD, SVC, SYP, SZL, THB, TJR, TMM, TND, TOP, TPE, TRL, TTD, TWD, TZS, UAH, UGX, USD, USN, USS, UYU, UZS, VEB, VND, VUV, WST, XAF, XAG, XAU, XBA, XBB, XBC, XBD, XCD, XDR, XFO, XFU, XOF, XPD, XPF, XPT, XTS, XXX, YER, YUM, ZAL, ZAR, ZMK, ZRN, ZWD.

XXX is used for transactions where no currency is involved.

11.1.8 TpChargingEventInfo

Defines the structure of data elements that specifies charging event information.

Structured Member Name	Structured Member Type	Description
ChargingEventName	TpChargingEventName	The charging event for which notifications can be requested by the application.
CurrentBalanceInfo	TpBalanceInfo	The current balance of the user's account.
ChargingEventTime	TpTime	The time at which the charging event occurred.

11.1.9 TpChargingEventCriteria

Defines the structure of data elements that specifies charging event criteria.

Structured Member Name	Structured Member Type	Description
ChargingEvents	TpChargingEventNameSet	Specifies the specific charging event criteria used by the application to define the event required.
Users	TpAddressSet	Specifies the user(s) for which the charging events are requested to be reported.

11.1.10 TpChargingEventNameSet

Defines a collection of TpChargingEventName elements.

11.1.11 TpChargingEventCriteriaResult

Defines the Sequence of Data Elements that specify the criteria relating to event requests.

Sequence Element Name	Sequence Element Type
ChargingEventCriteria	TpChargingEventCriteria
AssignmentID	TpAssignmentID

11.1.12 TpChargingEventCriteriaResultSet

Defines a collection of TpChargingEventCriteriaResult elements.

11.1.13 TpBalance

Defines the structure of data elements that specifies a balance.

Structured Member Name	Structured Member Type	Description
UserID	TpAddress	Specifies the user to whom the account belongs.
StatusCode	TpBalanceQueryError	Specifies the status code for the balance query request.
BalanceInfo	TpBalanceInfo	Specifies the balance information for the user.

11.1.14 TpBalanceSet

Defines a collection of TpBalance elements.

11.1.15 TpTransactionHistory

This data type is a sequence of data elements that describes the transaction history.

Sequence Element Name	Sequence Element Type	Description
TransactionID	TpAssignmentID	Specifies the ID of the specific transaction.
TimeStamp	TpDateAndTime	Specifies the date and time when the specific transaction was processed.
AdditionalInfo	TpString	Specifies a free format string providing additional information on the specific transaction. This could be the applicationDescription provided with the actual transaction.

11.1.16 TpTransactionHistorySet

Defines a collection of TpTransactionHistory elements.

11.1.17 TpTransactionHistoryStatus

Defines a status code that is reported by the Account Manager service capability feature as a result of a transaction history retrieval request.

Name	Value	Description
P_AM_TRANSACTION_ERROR_UNSPECIFIED	0	General error, unspecified.
P_AM_TRANSACTION_INVALID_INTERVAL	1	An invalid interval for the transaction history was specified.
P_AM_TRANSACTION_UNKNOWN_ACCOUNT	2	No account for the specified user is known.
P_AM_TRANSACTION_UNAUTHORIZED_APPLICATION	3	Application is not authorized to query balance.
P_AM_TRANSACTION_PROCESSING_ERROR	4	A processing error occurred while compiling the transaction history.
P_AM_TRANSACTION_SYSTEM_FAILURE	5	System failure. The request could not be handled.

11.1.18 TpBalanceExpiryDate

Defines the structure of data elements that specifies a balance expiry date.

Structure Element Name	Structure Element Type	Structure Element Description
UserID	TpAddress	Specifies the user to whom the account belongs.
StatusCode	TpBalanceQueryError	Specifies the status code for the balance query request.
ExpiryDate	TpDateAndTime	Defines the expiry date and time.

11.1.19 TpBalanceExpiryDateSet

Defines a Numbered set of data elements of TpBalanceExpiryDate.

11.1.20 TpVoucherError

Defines an error that is reported as a result of a voucher request.

Name	Value	Description
P_VOUCHER_OK	0	No error occurred while processing the request
P_VOUCHER_UNDEFINED	1	General error, unspecified
P_VOUCHER_UNKNOWN_SUBSCRIBER	2	Subscriber is unknown
P_VOUCHER_UNAUTHORIZED_APPLICATION	3	Application is not authorized
P_VOUCHER_SYSTEM_FAILURE	4	System failure. The request could not be handled

11.1.21 TpVoucher

Defines the structure of data elements that specifies a voucher.

Structure Element Name	Structure Element Type	Structure Element Description
VoucherID	TpAssignmentID	Specifies the unique identifier for this voucher.
UserID	TpAddress	Specifies the user to who the voucher belongs.
BalanceInfo	TpBalanceInfo	Specifies the amount of the voucher.

11.1.22 TpVoucherSet

Defines a numbered set of data elements of TpVoucher.

12 Exception Classes

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

Name	Description
P_UNAUTHORIZED_APPLICATION	Application is not authorized to perform charging operations.

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 Account Management SCF

The OMG IDL representation of this interface specification is contained in a text file (am.idl contained in archive es_20391511v010101p0.zip) which accompanies the present document.

Annex B (informative): W3C WSDL Description of Account Management SCF

The W3C WSDL representation of this interface specification is contained in a text file (am.wsdl contained in archive es_20391511v010101p0.zip) which accompanies the present document.

Annex C (informative): Java™ API Description of the Account Management SCF

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 20391511J2SE.zip.

The J2EE™ representation of this interface specification is provided as Java™ Code, contained in archive 20391511J2EE.zip.

Both these archives can be found in es_20391511v010101p0.zip which accompanies the present document.

Annex D (informative): Contents of 3GPP OSA Rel-6 Account Management

All of the present document is relevant for TS 129 198-11 V6 (Release 6).

Annex E (informative): Description of Account 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: Account 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: Account 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: Account Management Service Properties

There are no additions or exclusions.

E.2.11 Clause 11: Data Definitions

There are no additions or exclusions.

E.2.12 Clause 12: Exception Classes

There are no additions or exclusions.

E.2.13 Annex A (normative): OMG IDL Description of Account Management SCF

There are no additions or exclusions.

E.2.14 Annex B (informative): W3C WSDL Description of Account 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
Interfaces added in ES 203 915-11 version 1.1.1 (Parlay 5.0)	

F.1.2 Deprecated

Identifier	Comments
Interfaces deprecated in ES 203 915-11 version 1.1.1 (Parlay 5.0)	

F.1.3 Removed

Identifier	Comments
Interfaces removed in ES 203 915-11 version 1.1.1 (Parlay 5.0)	

F.2 Methods

F.2.1 New

Identifier	Comments
Methods added in ES 203 915-11 version 1.1.1 (Parlay 5.0)	
IpAccountManager.queryBalanceExpiryDateReq	
IpAccountManager.updateBalanceReq	
IpAccountManager.createVoucherReq	
IpAccountManager.queryVoucherReq	
IpAccountManager.destroyVoucherReq	
IpAccountManager.queryUserVouchersReq	
IpAppAccountManager.queryBalanceExpiryDateRes	
IpAppAccountManager.queryBalanceExpiryDateErr	
IpAppAccountManager.updateBalanceRes	
IpAppAccountManager.updateBalanceErr	
IpAppAccountManager.createVoucherRes	
IpAppAccountManager.createVoucherErr	
IpAppAccountManager.destroyVoucherRes	
IpAppAccountManager.destroyVoucherErr	
IpAppAccountManager.queryVoucherRes	
IpAppAccountManager.queryVoucherErr	
IpAppAccountManager.queryUserVouchersRes	
IpAppAccountManager.queryUserVouchersErr	

F.2.2 Deprecated

Identifier	Comments
Methods deprecated in ES 203 915-11 version 1.1.1 (Parlay 5.0)	

F.2.3 Modified

Identifier	Comments
Methods modified in ES 203 915-11 version 1.1.1 (Parlay 5.0)	

F.2.4 Removed

Identifier	Comments
Methods removed in ES 203 915-11 version 1.1.1 (Parlay 5.0)	

F.3 Data Definitions

F.3.1 New

Identifier	Comments
Data Definitions added in ES 203 915-11 version 1.1.1 (Parlay 5.0)	
TpBalanceExpiryDate	
TpBalanceExpiryDateSet	
TpVoucherError	
TpVoucher	
TpVoucher	

F.3.2 Modified

Identifier	Comments
Data Definitions modified in ES 203 915-11 version 1.1.1 (Parlay 5.0)	

F.3.3 Removed

Identifier	Comments
Data Definitions removed in ES 203 915-11 version 1.1.1 (Parlay 5.0)	

F.4 Service Properties

F.4.1 New

Identifier	Comments
Service Properties added in ES 203 915-11 version 1.1.1 (Parlay 5.0)	
P_NOTIFICATION_ADDRESS_RANGES	Replaces P_TRIGGERING_ADDRESSES

F.4.2 Deprecated

Identifier	Comments
Service Properties deprecated in ES 203 915-11 version 1.1.1 (Parlay 5.0)	

F.4.3 Modified

Identifier	Comments
Service Properties modified in ES 203 915-11 version 1.1.1 (Parlay 5.0)	

F.4.4 Removed

Identifier	Comments
Service Properties removed in ES 203 915-11 version 1.1.1 (Parlay 5.0)	
P_TRIGGERING_ADDRESSES	Replaced with P_NOTIFICATION_ADDRESS_RANGES

F.5 Exceptions

F.5.1 New

Identifier	Comments
Exceptions added in ES 203 915-11 version 1.1.1 (Parlay 5.0)	

F.5.2 Modified

Identifier	Comments
Exceptions modified in ES 203 915-11 version 1.1.1 (Parlay 5.0)	

F.5.3 Removed

Identifier	Comments
Exceptions removed in ES 203 915-11 version 1.1.1 (Parlay 5.0)	

F.6 Others

None.

History

Document history		
V1.1.1	February 2005	Membership Approval Procedure MV 20050408: 2005-02-08 to 2005-04-08
V1.1.1	April 2005	Publication