**3GPP TSG CN Plenary Meeting #16** **NP-020183**
**5th - 7th June 2002. Marco Island, USA.**

**Source:** **CN5 (OSA)**

**Title:** **Rel-5 CRs 29.198-xy (OSA API) Addition of Support for Network Controlled Notifications**

**Agenda item:** **8.2**

**Document for:** **APPROVAL**

---

| Doc-1st -Level | Spec | CR | Rv | Pha | Subject | Cat | Ver Curr | Ver New | Doc-2nd -Level | Work item |
|---|---|---|---|---|---|---|---|---|---|---|
| NP-020183 | 29.198-04 | 040 | - | Rel-5 | Addition of support for Network Controlled Notifications MPCC | B | 4.4.0 | 5.0.0 | N5-020454 | OSA2 |
| NP-020183 | 29.198-05 | 014 | - | Rel-5 | Addition of Support for Network Controlled Notifications UI | B | 4.4.0 | 5.0.0 | N5-020489 | OSA2 |
| NP-020183 | 29.198-08 | 007 | - | Rel-5 | Addition of Support for Network Controlled Notifications DSC | B | 4.4.0 | 5.0.0 | N5-020490 | OSA2 |
| NP-020183 | 29.198-11 | 010 | - | Rel-5 | Addition of Support for Network Controlled Notifications AM | B | 4.3.0 | 5.0.0 | N5-020491 | OSA2 |

**joint API group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5)** **N5-020454**
**Meeting #18, Budapest, HUNGARY, 13 – 17 May 2002**

CR-Form-v5

# CHANGE REQUEST

⌘ **29.198-04** **CR** **040** ⌘ **rev** **-** ⌘ Current version: **4.4.0** ⌘

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

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network ☒

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Addition of support for Network Controlled Notifications MPCC |
| ***Source:*** | ⌘ | CN5 |
| ***Work item code:*** | ⌘ | OSA2 |
| ***Date:*** | ⌘ | 17/05/2002 |
| ***Category:*** | ⌘ | **B** |
| ***Release:*** | ⌘ | REL-5 |

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

*Use one of the following releases:*
*2 (GSM Phase 2)*
*R96 (Release 1996)*
*R97 (Release 1997)*
*R98 (Release 1998)*
*R99 (Release 1999)*
*REL-4 (Release 4)*
*REL-5 (Release 5)*

| | | |
|---|---|---|
| ***Reason for change:*** | ⌘ | Addition of Support for Network Controlled Notifications MPCC is needed to satisfy new requirements in Rel-5. |
| ***Summary of change:*** | ⌘ | Addition of 2 new methods to Multi Party Call Control Manager to enable/disable reception of notifications by an application. |
| ***Consequences if not approved:*** | ⌘ | |

| | | | | |
|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 7.3.1, 7.4.1.3 | | |
| ***Other specs affected:*** | ⌘ | ☒ Other core specifications<br>☐ Test specifications<br>☐ O&M Specifications | ⌘ | 29.198-05, 29.198-08, 29.198-11 |
| ***Other comments:*** | ⌘ | | | |

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

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

# Introduction

At the moment OSA/Parlay applications have to explicitly request for triggers and notifications in the network. This means that the ASP is responsible for provisioning.

Another option is that provisioning of triggers is done in the Home Environment (by the Network Operator) and that an application only has to indicate it's availability and tell the SCS that it can receive notifications.

For this SA2 has stated the following requirements for OSA Release 5 (see 3GPP TS 23.127, Virtual Home Environment/Open Service Access):

> *Specific methods shall be specified in OSA Network Service Capability Features, permitting:*
>
> *(1) An OSA application to request user related event notifications pertaining to any subscribed user for which the service implemented by the application is activated.*
>
> *(2) The OSA SCS to report user related event notifications in which it explicitly identifies the user to which the event applies.*
>
> *(3) An OSA application to request a function to be applied to all current subscribed users for which the service implemented by the application is activated.*

# Proposed Changes

## Additional Methods

Ericsson proposes to add the following methods to a Service Manager that has to deal with notifications:

- enableNotifications()
  With this method the application indicates it is able to receive notifications for events in the network. The provisioning of these notifications has been done from within the Home Environment. This method does NOT influence notifications created with createNotification().

- disableNotifications()
  After this method is called, the application will no longer receive notifications for events that are created in the VHE. This method does NOT influence notifications created with createNotification().

Both these methods contain an assignment ID which is also used in reportNotification(). If an application uses both mechanisms (network provisioned and service provider provisioned) in parallel, this ID can be used to distinguish the kind of notification.

The new methods will be added to:

- IpMultiPartyCallControlManager
- IpUIManager
- IpDataSessionControlManager
- IpAccountManager

This contribution addresses only the impacts to Multi Party Call Control.

## Interworking with createNotification() and Handling Overlapping Criteria

By including enableNotifications()/disableNotifications() there are two mechanisms for dealing with notifications. These two mechanisms do not influence eachother except when criteria overlap. If notifications are created using createNotification() which are already provisioned in the network, an exception shall be raised (similar as yhe current way createNotification() is supposed to work). This ALSO applies to cases where the same application is involved. Note however that disableNotifications() does not influence reporting notifications requested with createNotification(). These can only be disbaled by calling destroyNotification().

## Callback Interface Used

To align enableNotifications() with createNotification(), the method carries a parameter (except for Account Management) indicating the call-back interface to be used. A value of NULL for this parameter indicates that the default call-back (set using setCallback()) is to be used. Calling enableNotifications() a second time with a different call-back interface, the second call-back interface is treated as an additional call-back that is used when calls to the initially provided call-back fail (e.g. due to overload or failure). As stated earlier, the assignment ID can be used to distinguish between the kind of notification (if needed).

## Alternatives

An alternative for adding the new methods would be to call createNotification() with "*" as address-range. For several reasons, this approach is not chosen:

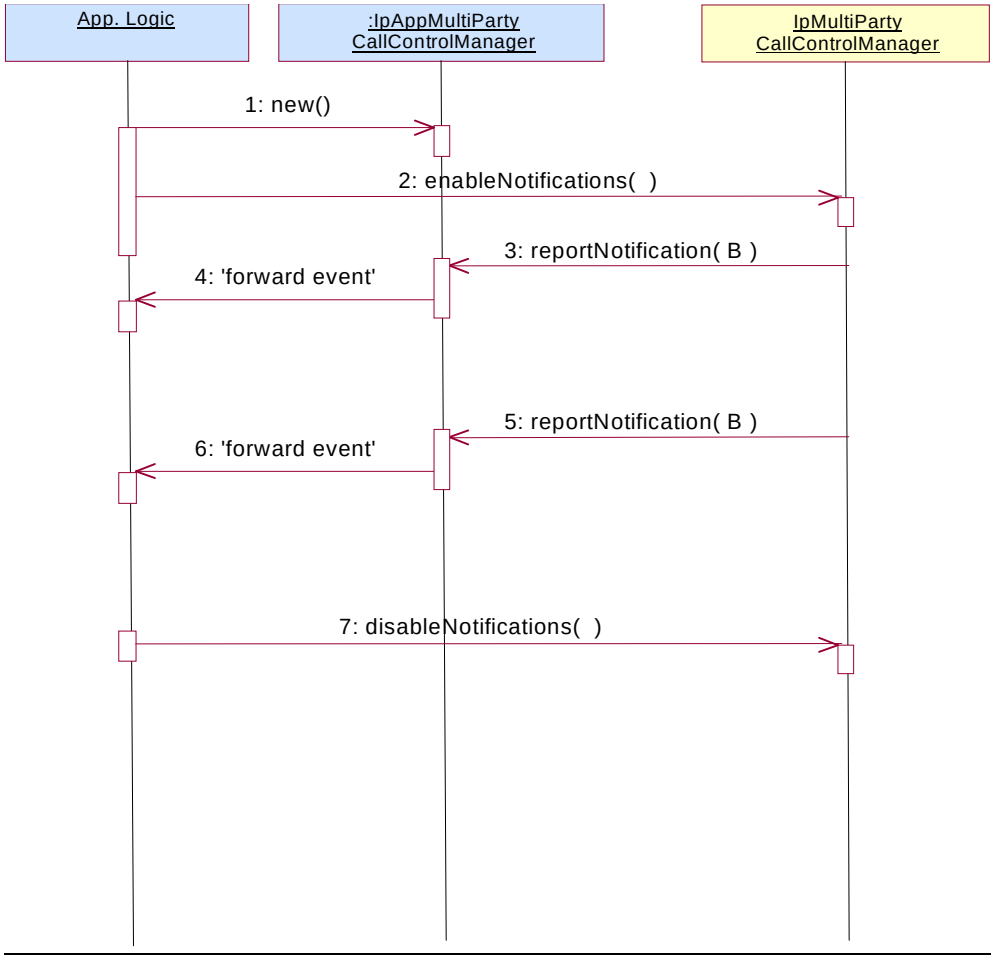
- createNotification() is used for other purposes (to explicitly request specific types of events for a specific address-range). Using the same method for indicating that an application is ready to receive any events might confuse application developers. Our view is that methods should be semantically well defined and serve a single purpose.
- createNotification() is mapped to MAP AnyTimeModification. For enableNotifications there is no mapping since the network is in control here. This is again an indication that the new intended functionality is different from createNotification and that therefore separate methods should be defined.

The impacts to 29.198-4 are listed below (with change *bars):*

## 7.1.1Sequence Diagrams

### 7.1.x 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 IpAppMultiPartyCallControlManager to handle callbacks.

2: The enableNotifications method is invoked on the IpMultiPartyCallControlManager 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.

## 7.3.1 Interface Class IpMultiPartyCallControlManager

| <<Interface>><br>IpMultiPartyCallControlManager |
|---|
| |
| createCall (appCall : in IpAppMultiPartyCallRef) : TpMultiPartyCallIdentifier<br>createNotification (appCallControlManager : in IpAppMultiPartyCallControlManagerRef, notificationRequest : in TpCallNotificationRequest) : TpAssignmentID<br>destroyNotification (assignmentID : in TpAssignmentID) : void<br>changeNotification (assignmentID : in TpAssignmentID, notificationRequest : in TpCallNotificationRequest) : void<br>getNotification () : TpNotificationRequestedSet<br>setCallLoadControl (duration : in TpDuration, mechanism : in TpCallLoadControlMechanism, treatment : in TpCallTreatment, addressRange : in TpAddressRange) : TpAssignmentID<br>enable~~Preset~~Notifications (appCallControlManager : in IpAppMultiPartyCallControlManagerRef) : TpAssignmentID<br>disable~~Preset~~Notifications (~~assignmentID : in TpAssignmentID~~) : void |

*Method*

### `createNotification()`

This method is used to enable call notifications so that events can be sent to the application. This is the first step an application has to do to get initial notifications of calls happening in the network. When such an event happens, the application will be informed by reportNotification(). In case the application is interested in other events during the context of a particular call session it has to use the createAndRouteCallLegReq() method on the call object or the eventReportReq() method on the call leg object. The application will get access to the call object when it receives the reportNotification(). (Note that createNotification() is not applicable if the call is setup by the application).

The createNotification method is purely intended for applications to indicate their interest to be notified when certain call events take place. It is possible to subscribe to a certain event for a whole range of addresses, e.g. the application can indicate it wishes to be informed when a call is made to any number starting with 800.

If some application already requested notifications with criteria that overlap the specified criteria or the specified criteria overlap with criteria already present in the network (when provisioned from within the network), the request is refused with P_INVALID_CRITERIA. The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used.

If a notification is requested by an application with monitor mode set to notify, then there is no need to check the rest of the criteria for overlapping with any existing request as the notify mode does not allow control on a call to be passed over. Only one application can place an interrupt request if the criteria overlaps.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used. In case the enableCallNotification 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 call control manager interface for this newly-enabled event notification.

*Parameters*

**appCallControlManager : in IpAppMultiPartyCallControlManagerRef**

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.

**notificationRequest : in TpCallNotificationRequest**

Specifies the event specific criteria used by the application to define the event required. Only events that meet these criteria are reported. Examples of events are "incoming call attempt reported by network", "answer", "no answer", "busy". Individual addresses or address ranges may be specified for destination and/or origination.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE**

*Method*

## destroyNotification()

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

*Parameters*

**assignmentID : in TpAssignmentID**

Specifies the assignment ID given by the generic call control manager interface when the previous enableNotification() was called. If the assignment ID does not correspond to one of the valid assignment IDs, the exception P_INVALID_ASSIGNMENTID will be raised. If two callbacks have been registered under this assignment ID both of them will be disabled.

*Raises*

**TpCommonExceptions, P_INVALID_ASSIGNMENT_ID**

*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 generic call control manager interface for the event notification. If two callbacks have been registered under this assignment ID both of them will be changed.

**notificationRequest : in TpCallNotificationRequest**

Specifies the new set of event specific 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**

*Method*

## getNotification()

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

Returns notificationsRequested: Specifies the notifications that have been requested by the application.

*Parameters*

No Parameters were identified for this method

*Returns*

**TpNotificationRequestedSet**

*Raises*

**TpCommonExceptions**

*Method*

## enable~~Preset~~Notifications()

This method is used to indicate that the application is able to receive notifications ~~of network events. These notifications have been~~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 requests to enable notifications for a second time with a different IpAppMultiPartyCallControlManager reference (i.e. without first disabling them), the second callback will be treated as an additional callback. This means that the callback will only be used in cases when the first callback specified by the application is unable to handle the callEventNotify (e.g. due to overload or failure).

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 provisoned 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 ~~is~~ relate~~d~~s to notifications provisioned from within the network~~. Furthermore it is required as input for disablePresetfications().~~Repeated calls to enableNotifications() return the same assignment ID.

*Parameters*

**appCallControlmanager : in IpAppMultiPartyCallControlManagerRef**

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**

*Method*

## disable~~Preset~~Notifications()

This method is used to indicate that the application is not able to receive notifications ~~of network events~~ 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 ~~operator controlled~~such notifications are reported anymore.

*Parameters*

~~**assignmentID : in TpAssignmentID**~~

~~Specifies the ID returned by enablePresetfications().~~No parameters are identified for this method.

*Raises*

**TpCommonExceptions**

### 7.4.1.3 Overview of allowed methods

| **Call Control Manager State** | **Methods applicable** |
|---|---|
| Active | createCall,<br>createNotification,<br>destroyNotification,<br>changeNotification,<br>getNotification,<br>setCallLoadControl,<br>enable~~Preset~~Notifications,<br>disable~~Preset~~Notifications |
| Interrupted | GetNotification,<br>Enable~~Preset~~Notifications,<br>Disable~~Preset~~Notifications |

## 7.5.2 Service Property values for the CAMEL Service Environment.

Implementations of the MultiParty Call Control API relying on the CSE shall have the Service Properties outlined above set to the indicated values :

```
P_OPERATION_SET = {
“IpMultiPartyCallControlManager.createNotification”,
“IpMultiPartyCallControlManager.destroyNotification”,
“IpMultiPartyCallControlManager.changeNotification”,
“IpMultiPartyCallControlManager.getNotification”,
“IpMultiPartyCallControlManager.enableNotifications”,
“IpMultiPartyCallControlManager.disableNotifications”,
“IpMultiPartyCallControlManager.setCallLoadControl”
“IpMultiPartyCall.getCallLegs”,
“IpMultiPartyCall.createCallLeg”,
“IpMultiPartyCall.createAndRouteCallLegReq”,
“IpMultiPartyCall.release”,
“IpMultiPartyCall.deassignCall”,
“IpMultiPartyCall.getInfoReq”,
“IpMultiPartyCall.setChargePlan”,
“IpMultiPartyCall.setAdviceOfCharge”,
“IpMultiPartyCall.superviseReq”,
“IpCallLeg.routeReq”,
“IpCallLeg.eventReportReq”,
“IpCallLeg.release”,
“IpCallLeg.getInfoReq”,
“IpCallLeg.getCall”,
“IpCallLeg.continueProcessing”
}

P_TRIGGERING_EVENT_TYPES = {
P_CALL_EVENT_CALL_ATTEMPT,
P_CALL_EVENT_ADDRESS_COLLECTED,
P_CALL_EVENT_ADDRESS_ANALYSED,
P_CALL_EVENT_RELEASE,
}

P_DYNAMIC_EVENT_TYPES = {
P_CALL_EVENT_ANSWER,
P_CALL_EVENT_RELEASE
}

P_ADDRESS_PLAN = {
P_ADDRESS_PLAN_E164
}

P_UI_CALL_BASED = {
TRUE
}

P_UI_AT_ALL_STAGES = {
FALSE
}

P_MEDIA_TYPE = {
P_AUDIO
}

P_MAX_CALLLEGS_PER_CALL = {
0,
2
}

P_UI_CALLLEG_BASED = {
FALSE
}

P_MEDIA_ATTACH_EXPLICIT = {
FALSE
}
```

**joint API group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5)** **N5-020489**
**Meeting #18, Budapest, HUNGARY, 13 – 17 May 2002**

CR-Form-v5

# CHANGE REQUEST

⌘ **29.198-05** **CR** **014** ⌘ **rev** **-** ⌘ Current version: **4.4.0** ⌘

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

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network ☒

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Addition of Support for Network Controlled Notifications UI |
| ***Source:*** | ⌘ | CN5 |
| ***Work item code:*** | ⌘ | OSA2 |
| ***Date:*** | ⌘ | 17/05/2002 |
| ***Category:*** | ⌘ | **B** |
| ***Release:*** | ⌘ | REL-5 |

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

*Use one of the following releases:*
*2 (GSM Phase 2)*
*R96 (Release 1996)*
*R97 (Release 1997)*
*R98 (Release 1998)*
*R99 (Release 1999)*
*REL-4 (Release 4)*
*REL-5 (Release 5)*

| | | |
|---|---|---|
| ***Reason for change:*** | ⌘ | Addition of Support for Network Controlled Notifications UI is needed to satisfy new requirements in Rel-5. |
| ***Summary of change:*** | ⌘ | Addition of 2 new methods to User Interaction Manager to enable/disable reception of notifications by an application. |
| ***Consequences if not approved:*** | ⌘ | |

| | | | | |
|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 4, 8.1, 9.1, 9.1.2 | | |
| ***Other specs affected:*** | ⌘ | ☒ Other core specifications<br>☐ Test specifications<br>☐ O&M Specifications | ⌘ | 29.198-04, 29.198-08, 29.198-11 |
| ***Other comments:*** | ⌘ | | | |

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

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

# Introduction

At the moment OSA/Parlay applications have to explicitly request for triggers and notifications in the network. This means that the ASP is responsible for provisioning.

Another option is that provisioning of triggers is done in the Home Environment (by the Network Operator) and that an application only has to indicate it's availability and tell the SCS that it can receive notifications.

For this SA2 has stated the following requirements for OSA Release 5 (see 3GPP TS 23.127, Virtual Home Environment/Open Service Access):

> *Specific methods shall be specified in OSA Network Service Capability Features, permitting:*
>
> *(1) An OSA application to request user related event notifications pertaining to any subscribed user for which the service implemented by the application is activated.*
>
> *(2) The OSA SCS to report user related event notifications in which it explicitly identifies the user to which the event applies.*
>
> *(3) An OSA application to request a function to be applied to all current subscribed users for which the service implemented by the application is activated.*

# Proposed Changes

## Additional Methods

Ericsson proposes to add the following methods to a Service Manager that has to deal with notifications:

- enableNotifications()
  With this method the application indicates it is able to receive notifications for events in the network. The provisioning of these notifications has been done from within the Home Environment. This method does NOT influence notifications created with createNotification().

- disableNotifications()
  After this method is called, the application will no longer receive notifications for events that are created in the VHE. This method does NOT influence notifications created with createNotification().

Both these methods contain an assignment ID which is also used in reportNotification(). If an application uses both mechanisms (network provisioned and service provider provisioned) in parallel, this ID can be used to distinguish the kind of notification.

The new methods will be added to:

- IpMultiPartyCallControlManager
- IpUIManager
- IpDataSessionControlManager
- IpAccountManager

This contribution addresses only the impacts to Multi Party Call Control.

## Interworking with createNotification() and Handling Overlapping Criteria

By including enableNotifications()/disableNotifications() there are two mechanisms for dealing with notifications. These two mechanisms do not influence eachother except when criteria overlap. If notifications are created using createNotification() which are already provisioned in the network, an exception shall be raised (similar as yhe current way createNotification() is supposed to work). This ALSO applies to cases where the same application is involved. Note however that disableNotifications() does not influence reporting notifications requested with createNotification(). These can only be disbaled by calling destroyNotification().

## Callback Interface Used

To align enableNotifications() with createNotification(), the method carries a parameter (except for Account Management) indicating the call-back interface to be used. A value of NULL for this parameter indicates that the default call-back (set using setCallback()) is to be used. Calling enableNotifications() a second time with a different call-back interface, the second call-back interface is treated as an additional call-back that is used when calls to the initially provided call-back fail (e.g. due to overload or failure). As stated earlier, the assignment ID can be used to distinguish between the kind of notification (if needed).

## Alternatives

An alternative for adding the new methods would be to call createNotification() with "*" as address-range. For several reasons, this approach is not chosen:

- createNotification() is used for other purposes (to explicitly request specific types of events for a specific address-range). Using the same method for indicating that an application is ready to receive any events might confuse application developers. Our view is that methods should be semantically well defined and serve a single purpose.
- createNotification() is mapped to MAP AnyTimeModification. For enableNotifications there is no mapping since the network is in control here. This is again an indication that the new intended functionality is different from createNotification and that therefore separate methods should be defined.

*The impacts to 29.198-5 are listed below (with change bars):*

# 4 Generic and Call User Interaction SCF

The Generic User Interaction service capability feature is used by applications to interact with end users. It consists of two interfaces:

– User Interaction Manager, containing management functions for User Interaction related issues;

– Generic User Interaction, containing methods to interact with an end-user.

The Generic User Interaction service capability feature is described in terms of the methods in the Generic User Interaction interfaces.

The following table gives an overview of the Generic User Interaction methods and to which interfaces these methods belong.

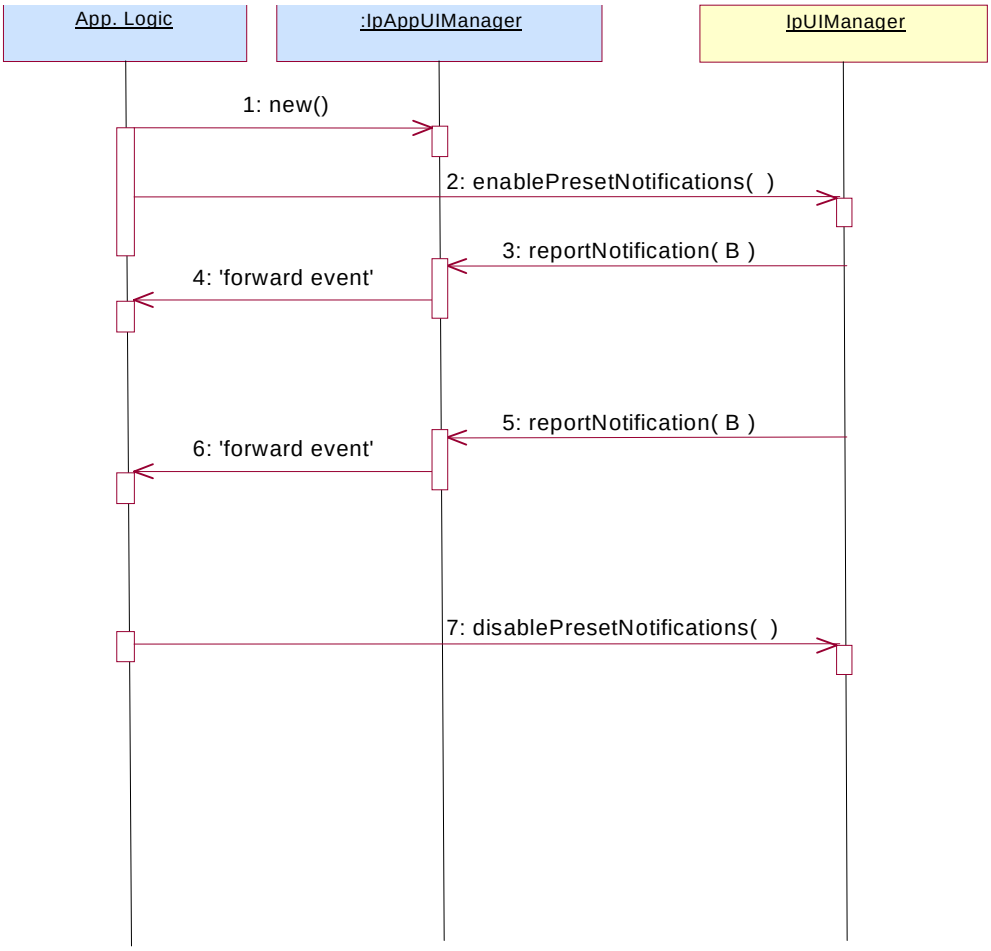
**Table 1: Overview of Generic User Interaction interfaces and their methods**

| User Interaction Manager | Generic User Interaction |
|---|---|
| createUI | sendInfoReq |
| createUICall | sendInfoRes |
| createNotification | sendInfoErr |
| destroyUINotification | sendInfoAndCollectReq |
| reportNotification | sendInfoAndCollectRes |
| userInteractionAborted | sendInfoAndCollectErr |
| userInteractionNotificationInterrupted | release |
| userInteractionNotificationContinued | UserInteractionFaultDetected |
| changeNotification | |
| getNotification | |
| enable~~Preset~~Notifications | |
| disable~~Preset~~Notifications | |

## 7.1 Sequence Diagrams

### 7.1.x 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 IpAppUIManager to handle callbacks.

2: The enable~~Preset~~Notifications method is invoked on the IpUIManager 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.

~~6~~5: When a network created trigger occurs the application is notified on the callback interface.

~~7~~6: The event is forwarded to the application.

~~10~~7: When the application does not want to receive notifications created in the network anymore, it invokes disable~~Preset~~Notifications on the IpMultiPartyCallConrolManager interface. From now on the gateway will not send any notifications to the application that are created in the network

## 8.1 Interface Class IpUIManager

| <<Interface>><br>IpUIManager |
| --- |
| |

createUI (appUI : in IpAppUIRef, userAddress : in TpAddress) : TpUIIdentifier

createUICall (appUI : in IpAppUICallRef, uiTargetObject : in TpUITargetObject) : TpUICallIdentifier

createNotification (appUIManager : in IpAppUIManagerRef, eventCriteria : in TpUIEventCriteria) : TpAssignmentID

destroyNotification (assignmentID : in TpAssignmentID) : void

changeNotification (assignmentID : in TpAssignmentID, evenCriteria : in TpUIEventCriteria) : void

getNotification () : TpUIEventCriteriaResultSet

enable~~Preset~~Notifications (appUIManager : in IpAppUIManagerRef) : TpAssignmentID

disable~~Preset~~Notifications (~~assignmentID : in TpAssignmentID~~) : void

*Method*

## createNotification()

This method is used by the application to install specified notification criteria, for which the reporting is implicitly activated. If some application already requested notifications with criteria that overlap the specified criteria or the specified criteria overlap with criteria already present in the network (when provisioned from within the network), the request is refused with P_INVALID_CRITERIA.

The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used and the same servicecode is used.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. This means that the callback will only be used in case when the first callback specified by the application is unable to handle the reportNotification (e.g., due to overload or failure).

Returns: assignmentID

Specifies the ID assigned by the generic user interaction manager interface for this newly installed notification criteria.

*Parameters*

**appUIManager : in IpAppUIManagerRef**

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 TpUIEventCriteria**

Specifies the event specific criteria used by the application to define the event required, like user address and service code.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE**

*Method*

## destroyNotification()

This method is used by the application to destroy previously installed notification criteria via the createNotification method.

*Parameters*

**assignmentID : in TpAssignmentID**

Specifies the assignment ID given by the generic user interaction manager interface when the previous createNotification() 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_ASSIGNMENT_ID.

*Raises*

**TpCommonExceptions, P_INVALID_ASSIGNMENT_ID**

*Method*

## changeNotification()

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

*Parameters*

**assignmentID : in TpAssignmentID**

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

**eventCriteria : in TpUIEventCriteria**

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**

*Method*

## getNotification()

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

Returns: eventCriteria

Specifies the event specific 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*

**TpUIEventCriteriaResultSet**

*Raises*

**TpCommonExceptions, P_INVALID_CRITERIA**

*Method*

## enable~~Preset~~Notifications()

This method is used to indicate that the application is able to receive notifications ~~of network events. These notifications have been~~ 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 requests to enable notifications for a second time with a different IpAppUIManager reference (i.e. without first disabling them), the second callback will be treated as an additional callback. This means that the callback wil only be used in cases when the first callback specified by the application is unable to handle the callEventNotify (e.g. due to overload or failure).

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 provisoned 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 ~~is~~ relate~~d~~s to notifications provisioned from within the network~~. Furthermore it is required as input for disablePresetNotifications().~~

*Parameters*

**appCallControlmanager : in IpAppUIManagerRef**

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

*Method*
## disable~~Preset~~Notifications()

This method is used to indicate that the application is not able to receive notifications ~~of network events~~ 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 ~~operator controlled~~ such notifications are reported anymore.

*Parameters*

~~**assignmentID : in TpAssignmentID**~~

~~Specifies the ID returned by enablePresetNotifications().~~**No parameters for this method are identified.**

*Raises*

**TpCommonExceptions**

# 9 State Transition Diagrams

## 9.1 State Transition Diagrams for IpUIManager

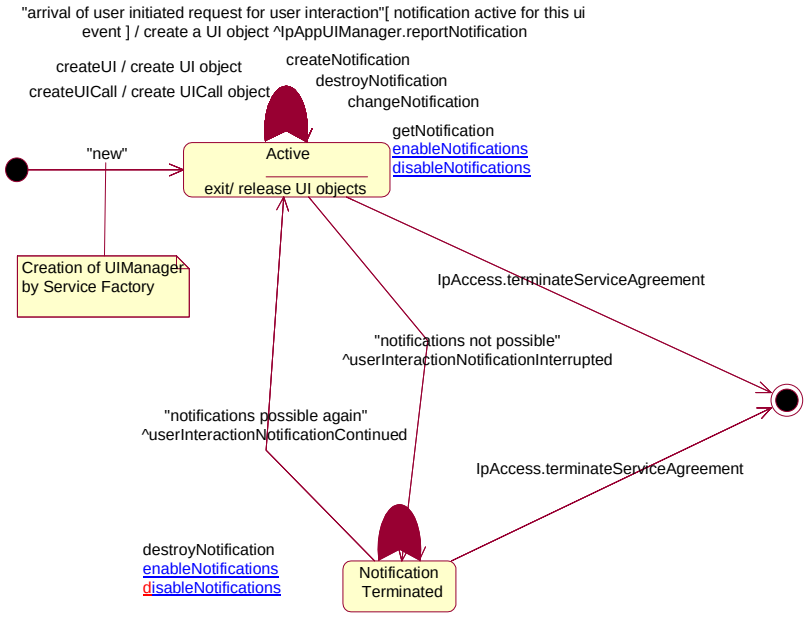



**Figure : Application view on the UI Manager**

### 9.1.1 Active State

In this state a relation between the Application and a User Interaction Service Capability Feature (Generic User Interaction or Call User Interaction) has been established. The application is now able to request creation of UI and/orUICall objects.

### 9.1.2 Notification Terminated State

When the UI manager is in the Notification terminated state, events requested with createNotification()/enable~~Preset~~Notifications() will not be forwarded to the application. There can be multiple reasons for this: for instance it might be that the application receives more notifications than defined in the Service Level Agreement. Another example is that the SCS has detected it receives no notifications from the network due to e.g. a link failure. In this state no requests for new notifications will be accepted.

**joint API group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5)** **N5-020490**
**Meeting #18, Budapest, HUNGARY, 13 – 17 May 2002**

CR-Form-v5

# CHANGE REQUEST

⌘ **29.198-08** **CR** **007** ⌘ **rev** **-** ⌘ Current version: **4.4.0** ⌘

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

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network ☒

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Addition of Support for Network Controlled Notifications DSC |
| ***Source:*** | ⌘ | CN5 |
| ***Work item code:*** | ⌘ | OSA2 |
| ***Date:*** | ⌘ | 17/05/2002 |
| ***Category:*** | ⌘ | **B** |
| ***Release:*** | ⌘ | REL-5 |

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

*Use one of the following releases:*
*2 (GSM Phase 2)*
*R96 (Release 1996)*
*R97 (Release 1997)*
*R98 (Release 1998)*
*R99 (Release 1999)*
*REL-4 (Release 4)*
*REL-5 (Release 5)*

| | | |
|---|---|---|
| ***Reason for change:*** | ⌘ | Addition of Support for Network Controlled Notifications DSC is needed to satisfy new requirements in Rel-5. |
| ***Summary of change:*** | ⌘ | Addition of 2 new methods to Data Session Control Manager to enable/disable reception of notifications by an application. |
| ***Consequences if not approved:*** | ⌘ | |

| | | | | |
|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 4, 8.4 | | |
| ***Other specs affected:*** | ⌘ | ☒ Other core specifications<br>☐ Test specifications<br>☐ O&M Specifications | ⌘ | 29.198-04, 29.198-05, 29.198-11 |
| ***Other comments:*** | ⌘ | | | |

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

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

# Introduction

At the moment OSA/Parlay applications have to explicitly request for triggers and notifications in the network. This means that the ASP is responsible for provisioning.

Another option is that provisioning of triggers is done in the Home Environment (by the Network Operator) and that an application only has to indicate it's availability and tell the SCS that it can receive notifications.

For this SA2 has stated the following requirements for OSA Release 5 (see 3GPP TS 23.127, Virtual Home Environment/Open Service Access):

> *Specific methods shall be specified in OSA Network Service Capability Features, permitting:*
>
> *(1) An OSA application to request user related event notifications pertaining to any subscribed user for which the service implemented by the application is activated.*
>
> *(2) The OSA SCS to report user related event notifications in which it explicitly identifies the user to which the event applies.*
>
> *(3) An OSA application to request a function to be applied to all current subscribed users for which the service implemented by the application is activated.*

# Proposed Changes

## Additional Methods

Ericsson proposes to add the following methods to a Service Manager that has to deal with notifications:

- enableNotifications()
  With this method the application indicates it is able to receive notifications for events in the network. The provisioning of these notifications has been done from within the Home Environment. This method does NOT influence notifications created with createNotification().

- disableNotifications()
  After this method is called, the application will no longer receive notifications for events that are created in the VHE. This method does NOT influence notifications created with createNotification().

Both these methods contain an assignment ID which is also used in reportNotification(). If an application uses both mechanisms (network provisioned and service provider provisioned) in parallel, this ID can be used to distinguish the kind of notification.

The new methods will be added to:

- IpMultiPartyCallControlManager
- IpUIManager
- IpDataSessionControlManager
- IpAccountManager

This contribution addresses only the impacts to Multi Party Call Control.

## Interworking with createNotification() and Handling Overlapping Criteria

By including enableNotifications()/disableNotifications() there are two mechanisms for dealing with notifications. These two mechanisms do not influence eachother except when criteria overlap. If notifications are created using createNotification() which are already provisioned in the network, an exception shall be raised (similar as yhe current way createNotification() is supposed to work). This ALSO applies to cases where the same application is involved. Note however that disableNotifications() does not influence reporting notifications requested with createNotification(). These can only be disbaled by calling destroyNotification().

## Callback Interface Used

To align enableNotifications() with createNotification(), the method carries a parameter (except for Account Management) indicating the call-back interface to be used. A value of NULL for this parameter indicates that the default call-back (set using setCallback()) is to be used. Calling enableNotifications() a second time with a different call-back interface, the second call-back interface is treated as an additional call-back that is used when calls to the initially provided call-back fail (e.g. due to overload or failure). As stated earlier, the assignment ID can be used to distinguish between the kind of notification (if needed).

## Alternatives

An alternative for adding the new methods would be to call createNotification() with "*" as address-range. For several reasons, this approach is not chosen:

- createNotification() is used for other purposes (to explicitly request specific types of events for a specific address-range). Using the same method for indicating that an application is ready to receive any events might confuse application developers. Our view is that methods should be semantically well defined and serve a single purpose.
- createNotification() is mapped to MAP AnyTimeModification. For enableNotifications there is no mapping since the network is in control here. This is again an indication that the new intended functionality is different from createNotification and that therefore separate methods should be defined.

*The impacts to 29.198-8 are listed below (with change bars):*

# 4 Data Session Control SCF

The Data Session Control SCFs are described in terms of the methods in the Data Session Control interfaces. Table 1 gives an overview of the Data Session Control methods and to which interfaces these methods belong.

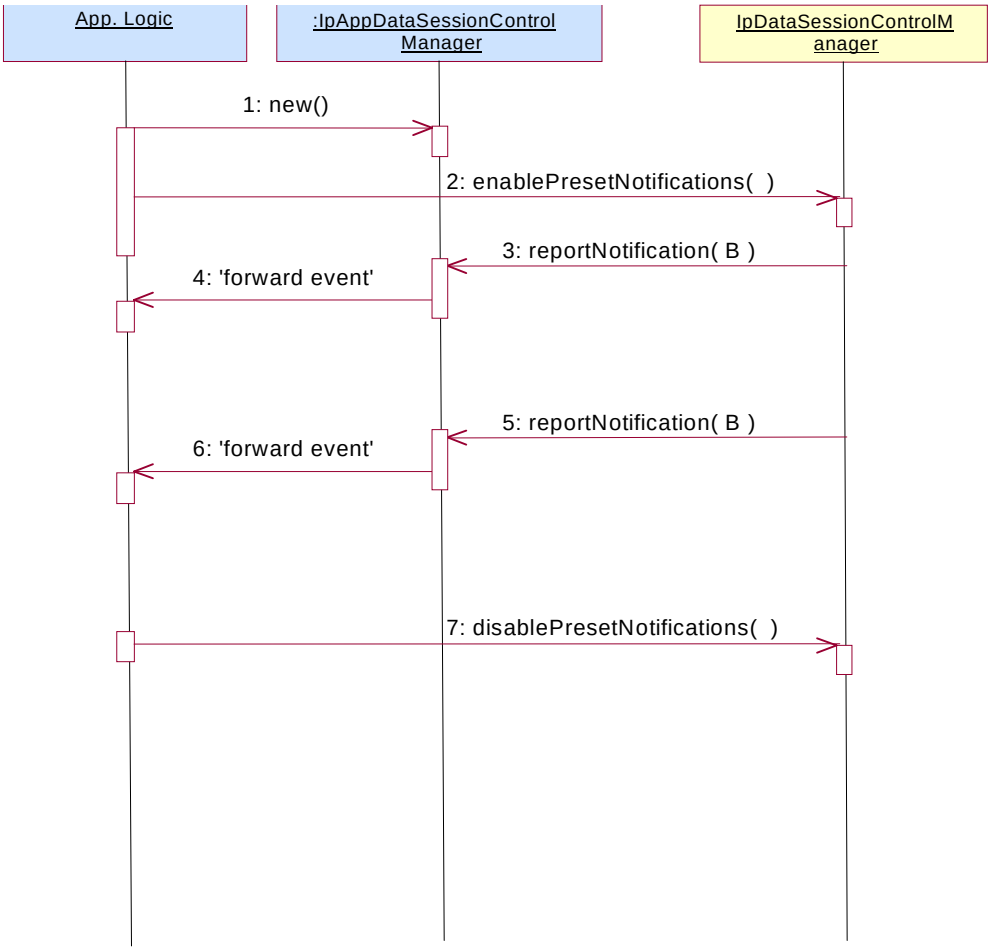
**Table 1: Overview of Data Session Control interfaces and their methods**

| Data Session Manager | Data Session |
|---|---|
| createNotification | connectReq |
| destroyNotification | connectRes |
| dataSessionNotificationInterrupted | connectErr |
| dataSessionNotificationContinued | release |
| reportNotification | superviseDataSessionReq |
| dataSessionAborted | superviseDataSessionRes |
| getNotification | superviseDataSessionErr |
| changeNotification | dataSessionFaultDetected |
| enableNotifications | setAdviceofCharge |
| disableNotifications | setDataSessionChargePlan |

## 7.1.1 Sequence Diagrams

## 7.1.x 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 IpAppDataSessionControlManager to handle callbacks.

2: The enableNotifications method is invoked on the IpDataSessionControlManager 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.

~~6~~5: When a network created trigger occurs the application is notified on the callback interface.

~~7~~6: The event is forwarded to the application.

~~10~~7: When the application does not want to receive notifications created in the network anymore, it invokes disableNotifications on the IpDataSessionConrolManager 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.

## *8.4* Interface Class IpDataSessionControlManager

| <<Interface>><br>IpDataSessionControlManager |
| --- |
| |
| createNotification (appDataSessionControlManager : in |

IpAppDataSessionControlManagerRef, eventCriteria : in TpDataSessionEventCriteria) : TpAssignmentID

destroyNotification (assignmentID : in TpAssignmentID) : void

changeNotification (assignmentID : in TpAssignmentID, eventCriteria : in TpDataSessionEventCriteria) : void

getNotification () : TpDataSessionEventCriteria

enable~~Preset~~Notifications (appDataSessionControlManager : in IpAppDataSessionControlManagerRef) : TpAssignmentID

disable~~Preset~~Notifications (~~assignmentID : in TpAssignmentID~~) : void

*Method*

## createNotification()

This method is used to enable data session notifications so that events can be sent to the application. This is the first step an application has to do to get initial notifications of data session happening in the network. When such an event happens, the application will be informed by reportNotification(). In case the application is interested in other events during the context of a particular data session it has to use the connectReq() method on the data session object. The application will get access to the data session object when it receives the reportNotification().

The createNotification method is purely intended for applications to indicate their interest to be notified when certain data session events take place. It is possible to subscribe to a certain event for a whole range of addresses, e.g. the application can indicate it wishes to be informed when a data session is setup to any number starting with 800.

If some application already requested notifications with criteria that overlap the specified criteria or the specified criteria overlap with criteria already present in the network (when provisioned from within the network), the request is refused with P_INVALID_CRITERIA. The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is 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 Data Session Manager object for this newly-enabled event notification.

*Parameters*

### appDataSessionControlManager : in IpAppDataSessionControlManagerRef

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 TpDataSessionEventCriteria

Specifies the event specific criteria used by the application to define the event required. Individual addresses or address ranges may be specified for destination and/or origination. Examples of events are "Data Session set up".

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P_SERVICE_INFORMATION_MISSING, P_SERVICE_FAULT_ENCOUNTERED, P_INVALID_NETWORK_STATE, P_INVALID_ADDRESS, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE**

*Method*

## destroyNotification()

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

*Parameters*

**assignmentID : in TpAssignmentID**

Specifies the assignment ID given by the data session manager object when the previous createNotification() was done.

*Raises*

**TpCommonExceptions, P_SERVICE_INFORMATION_MISSING, P_SERVICE_FAULT_ENCOUNTERED, P_INVALID_NETWORK_STATE, P_INVALID_ASSIGNMENT_ID**

*Method*

## changeNotification()

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

*Parameters*

**assignmentID : in TpAssignmentID**

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

**eventCriteria : in TpDataSessionEventCriteria**

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_SERVICE_INFORMATION_MISSING, P_SERVICE_FAULT_ENCOUNTERED, P_INVALID_NETWORK_STATE, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE**

*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 requirements are reported.

*Parameters*
No Parameters were identified for this method

*Returns*

**TpDataSessionEventCriteria**

*Raises*

**TpCommonExceptions, P_SERVICE_INFORMATION_MISSING, P_SERVICE_FAULT_ENCOUNTERED, P_INVALID_NETWORK_STATE**

*Method*
## enable~~Preset~~Notifications()

This method is used to indicate that the application is able to receive ~~notifications of network events. These notifications have been~~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 requests to enable notifications for a second time with a different IpAppDataSessionControlManager reference (i.e. without first disabling them), the second callback will be treated as an additional callback. This means that the callback wil only be used in cases when the first callback specified by the application is unable to handle the callEventNotify (e.g. due to overload or failure).

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 provisoned 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 ~~is~~ relate~~d~~s to notifications provisioned from within the network. ~~Furthermore it is required as input for disablePresetNotifications().~~ Repeated calls to enableNotifications() return the same assignment ID.

*Parameters*

**appCallControlmanager : in IpAppDataSessionControlManagerRef**

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

*Method*
## disable~~Preset~~Notifications()

This method is used to indicate that the application is not able to receive notifications ~~of network events~~ 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 ~~operator controlled~~such notifications are reported anymore.

*Parameters*

~~**assignmentID : in TpAssignmentID**~~

~~Specifies the ID returned by enablePresetNotifications().~~**No parameters for this method are identified.**

*Raises*

TpCommonExceptions

**joint API group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5)** **N5-020491**
**Meeting #18, Budapest, HUNGARY, 13 – 17 May 2002**

CR-Form-v5

# CHANGE REQUEST

⌘ **29.198-11** **CR** **010** ⌘ **rev** **-** ⌘ Current version: **4.3.0** ⌘

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

***Proposed change affects:*** ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network ☒

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Addition of Support for Network Controlled Notifications AM |
| ***Source:*** | ⌘ | CN5 |
| ***Work item code:*** | ⌘ | OSA2 |
| ***Date:*** | ⌘ | 17/05/2002 |
| ***Category:*** | ⌘ | **B** |
| ***Release:*** | ⌘ | REL-5 |

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

*Use one of the following releases:*
*2 (GSM Phase 2)*
*R96 (Release 1996)*
*R97 (Release 1997)*
*R98 (Release 1998)*
*R99 (Release 1999)*
*REL-4 (Release 4)*
*REL-5 (Release 5)*

| | | |
|---|---|---|
| ***Reason for change:*** | ⌘ | Addition of Support for Network Controlled Notifications AM is needed to satisfy new requirements in Rel-5. |
| ***Summary of change:*** | ⌘ | Addition of 2 new methods to Account Manager to enable/disable reception of notifications by an application. |
| ***Consequences if not approved:*** | ⌘ | |

| | | | | |
|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 8.1, 9.1, 9.1.1, 9.1.2 | | |
| ***Other specs affected:*** | ⌘ | ☒ Other core specifications<br>☐ Test specifications<br>☐ O&M Specifications | ⌘ | 29.198-04, 29.198-05, 29.198-08 |
| ***Other comments:*** | ⌘ | | | |

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

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

# Introduction

At the moment OSA/Parlay applications have to explicitly request for triggers and notifications in the network. This means that the ASP is responsible for provisioning.

Another option is that provisioning of triggers is done in the Home Environment (by the Network Operator) and that an application only has to indicate it's availability and tell the SCS that it can receive notifications.

For this SA2 has stated the following requirements for OSA Release 5 (see 3GPP TS 23.127, Virtual Home Environment/Open Service Access):

> *Specific methods shall be specified in OSA Network Service Capability Features, permitting:*
>
> *(1) An OSA application to request user related event notifications pertaining to any subscribed user for which the service implemented by the application is activated.*
>
> *(2) The OSA SCS to report user related event notifications in which it explicitly identifies the user to which the event applies.*
>
> *(3) An OSA application to request a function to be applied to all current subscribed users for which the service implemented by the application is activated.*

# Proposed Changes

## Additional Methods

Ericsson proposes to add the following methods to a Service Manager that has to deal with notifications:

- enableNotifications()
  With this method the application indicates it is able to receive notifications for events in the network. The provisioning of these notifications has been done from within the Home Environment. This method does NOT influence notifications created with createNotification().

- disableNotifications()
  After this method is called, the application will no longer receive notifications for events that are created in the VHE. This method does NOT influence notifications created with createNotification().

Both these methods contain an assignment ID which is also used in reportNotification(). If an application uses both mechanisms (network provisioned and service provider provisioned) in parallel, this ID can be used to distinguish the kind of notification.

The new methods will be added to:

- IpMultiPartyCallControlManager
- IpUIManager
- IpDataSessionControlManager
- IpAccountManager

This contribution addresses only the impacts to Multi Party Call Control.

## Interworking with createNotification() and Handling Overlapping Criteria

By including enableNotifications()/disableNotifications() there are two mechanisms for dealing with notifications. These two mechanisms do not influence eachother except when criteria overlap. If notifications are created using createNotification() which are already provisioned in the network, an exception shall be raised (similar as yhe current way createNotification() is supposed to work). This ALSO applies to cases where the same application is involved. Note however that disableNotifications() does not influence reporting notifications requested with createNotification(). These can only be disbaled by calling destroyNotification().

## Callback Interface Used

To align enableNotifications() with createNotification(), the method carries a parameter (except for Account Management) indicating the call-back interface to be used. A value of NULL for this parameter indicates that the default call-back (set using setCallback()) is to be used. Calling enableNotifications() a second time with a different call-back interface, the second call-back interface is treated as an additional call-back that is used when calls to the initially provided call-back fail (e.g. due to overload or failure). As stated earlier, the assignment ID can be used to distinguish between the kind of notification (if needed).

## Alternatives

An alternative for adding the new methods would be to call createNotification() with "*" as address-range. For several reasons, this approach is not chosen:

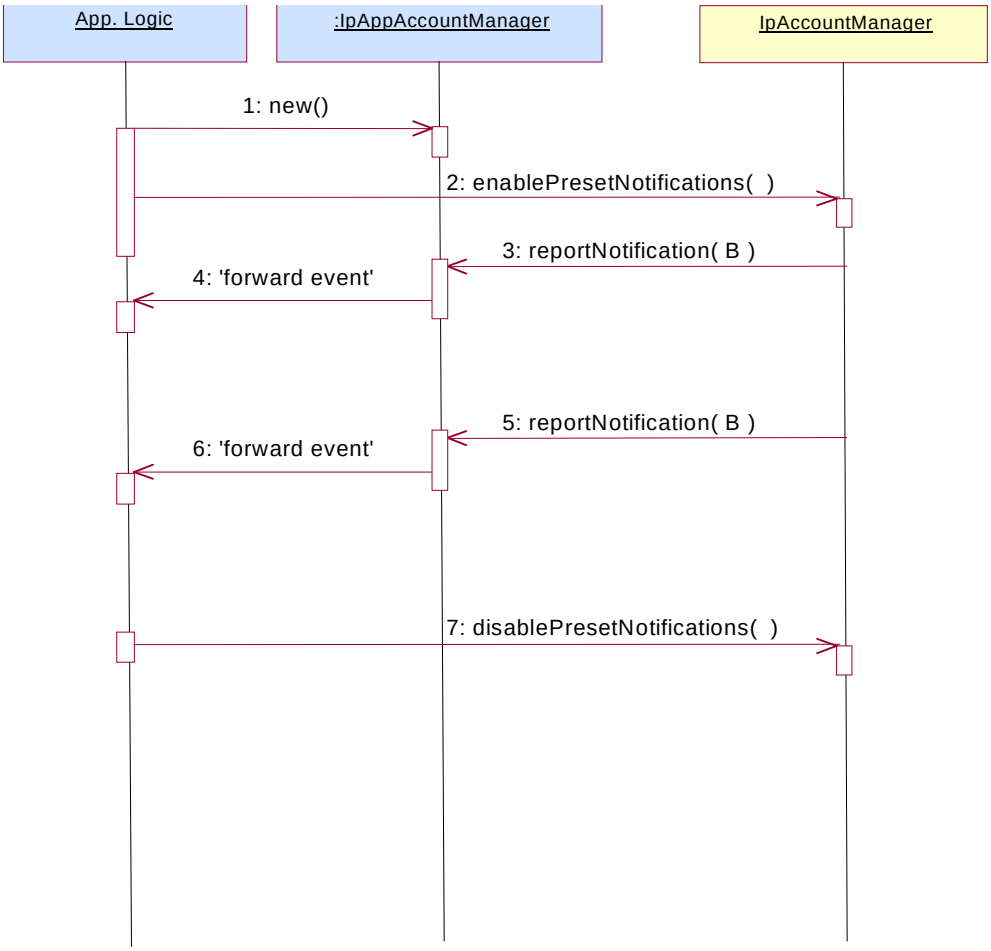
- createNotification() is used for other purposes (to explicitly request specific types of events for a specific address-range). Using the same method for indicating that an application is ready to receive any events might confuse application developers. Our view is that methods should be semantically well defined and serve a single purpose.
- createNotification() is mapped to MAP AnyTimeModification. For enableNotifications there is no mapping since the network is in control here. This is again an indication that the new intended functionality is different from createNotification and that therefore separate methods should be defined.

*The impacts to 29.198-11 are listed below (with change bars):*

## 7.1.1 Sequence Diagrams

### 7.1.x 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.

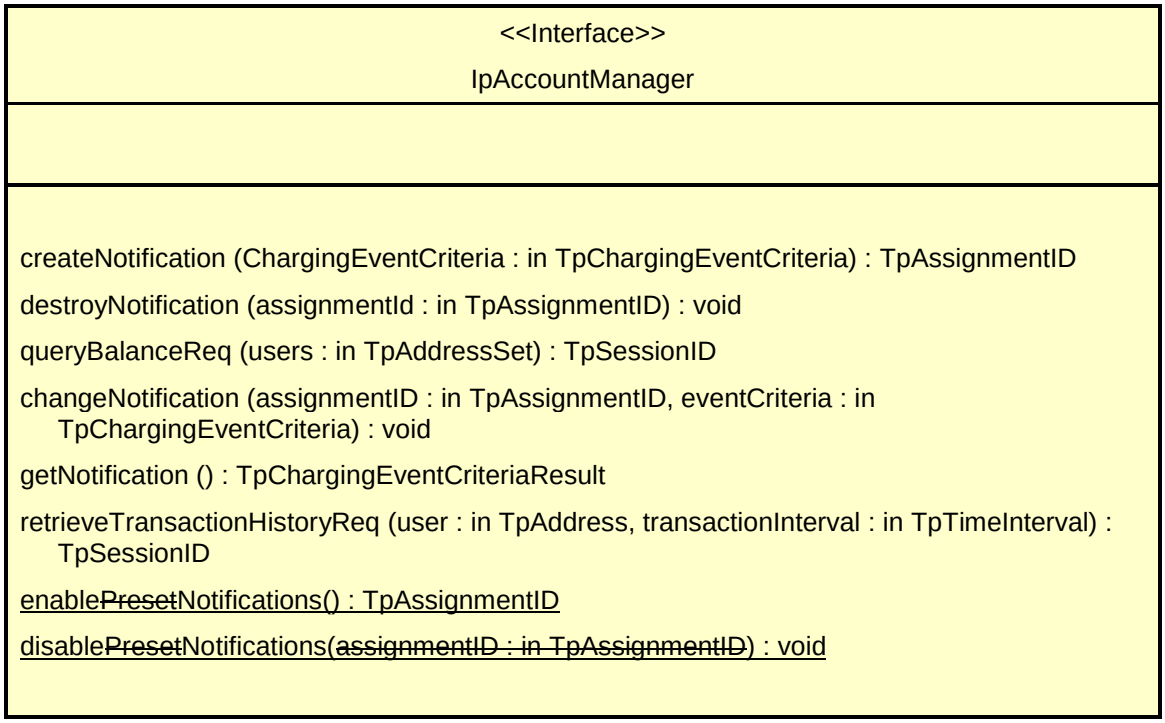
5: When a network created trigger occurs the application is notified on the callback interface.

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.

## *8.1* Interface Class IpAccountManager




*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`**

*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**

*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**

*Method*

## enable~~Preset~~Notifications()

This method is used to indicate that the application is able to receive ~~notifications of network events. These notifications have been~~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.

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 provisoned 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 ~~is~~ relate~~d~~s to notifications provisioned from within the network~~. Furthermore it is required as input for disablePresetNotifications().~~ Repeated calls to enableNotifications() return the same assignment ID.

*Parameters*

No parameters are identified for this method.

*Returns*

**TpAssignmentID**

*Raises*

TpCommonExceptions

*Method*

**disable~~Preset~~Notifications()**

This method is used to indicate that the application is not able to receive notifications ~~of network events~~ 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 ~~operator controlled~~such notifications are reported anymore.

*Parameters*

~~**assignmentID : in TpAssignmentID**~~

~~Specifies the ID returned by enablePresetNotifications().~~**No parameters are identified for this method.**

*Raises*

TpCommonExceptions

# 9 State Transition Diagrams

## 9.1 State Transition Diagrams for IpAccountManager

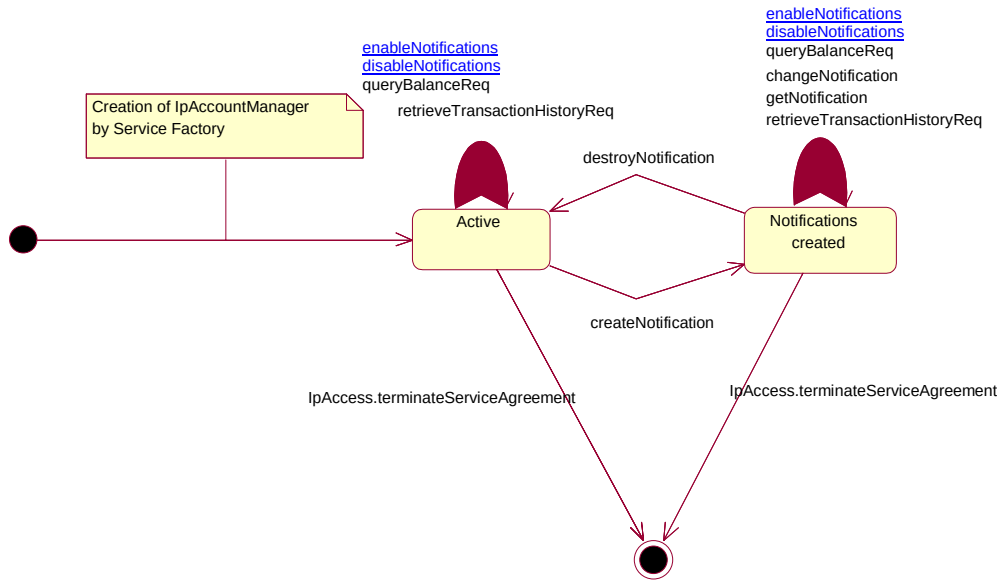



**Figure : Application view on the IpAccountManager**

### 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/enable~~Preset~~Notifications. 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~~()~~./disable~~Preset~~Notifications.

### 9.1.2 Notifications created State

When the Account Manager is in the Notifications created state, events requested with createNotification~~()~~ /enable~~Preset~~Notifications 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.