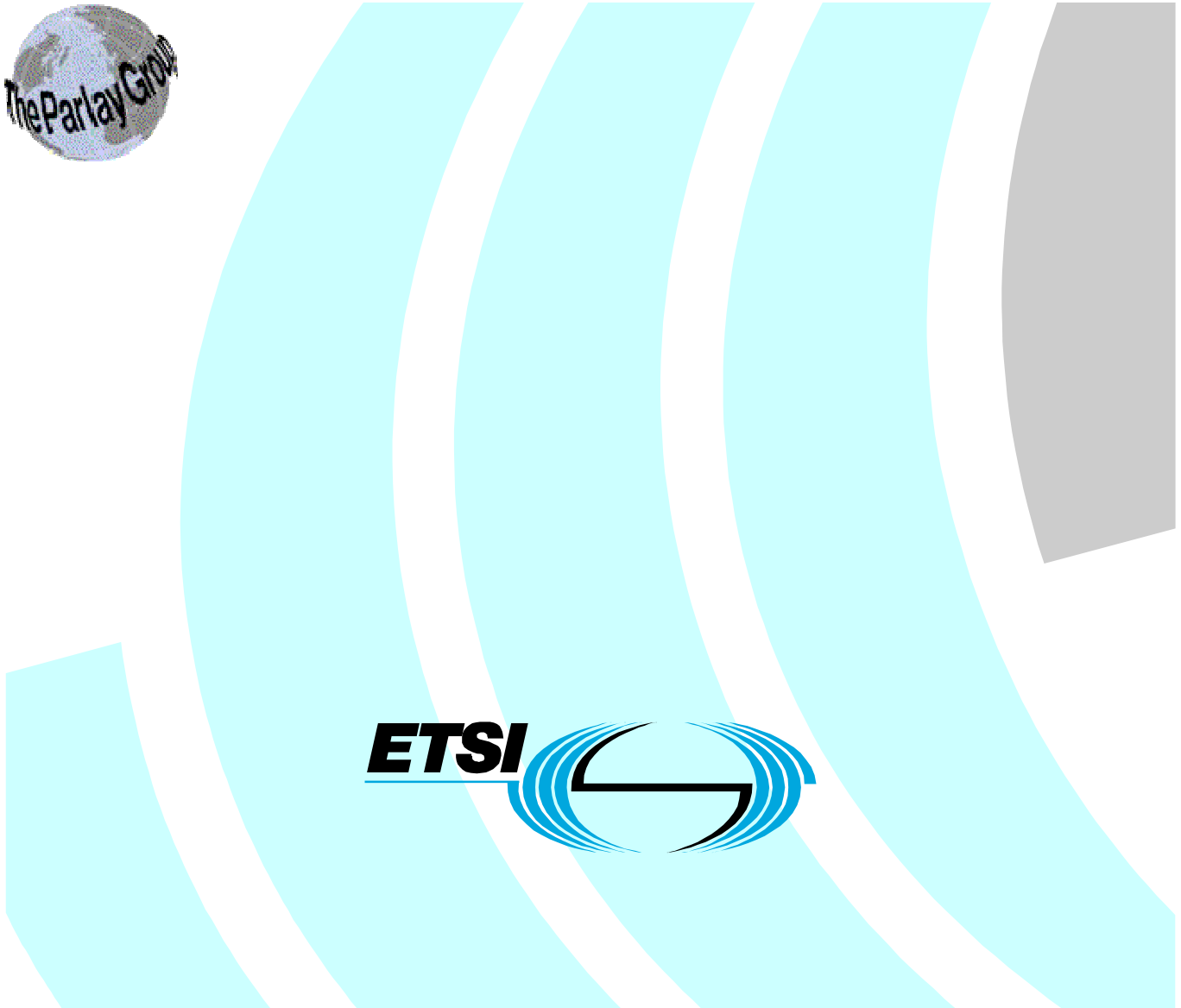
ETSI TR 102 397-10-2 V1.1.1 (2005-08)

Technical Report

Open Service Access (OSA);
Mapping of Parlay X Web Services to Parlay/OSA APIs;
Part 10: Call Handling Mapping;
Sub-part 2: Mapping to Multi-Party Call Control and User Interaction



Reference
DTR/TISPAN-01021-10-02-OSA

Keywords
API, OSA, service

ETSI

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

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

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

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

The present document is part 10, sub-part 2 of a multi-part deliverable covering Open Service Access (OSA); Mapping of Parlay X Web Services to Parlay/OSA APIs, as identified below:

Part 1: "Common Mapping";

Part 2: "Third Party Call Mapping";

Part 3: "Call Notification Mapping";

Part 4: "Short Messaging Mapping";

Part 5: "Multimedia Messaging Mapping";

Part 6: "Payment Mapping";

Part 7: "Account Management Mapping";

Part 8: "Terminal Status Mapping";

Part 9: "Terminal Location Mapping";

Part 10: "Call Handling Mapping";

Sub-part 1: "Mapping to Generic Call Control and User Interaction";

Sub-part 2: "Mapping to Multi-Party Call Control and User Interaction";

Part 11: "Audio Call Mapping";

Part 12: "Multimedia Conference Mapping";

Part 14: "Presence Mapping".

NOTE: Part 13 has not been provided as there is currently no defined mapping between ES 202 391-13 [4] and the Parlay/OSA APIs. If a mapping is developed, it will become part 13 of this series.

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

1 Scope

The present document specifies the mapping of the Parlay X Call Handling Web Service to the Multi-Party Call Control and User Interaction Service Capability Features (SCFs).

The Parlay X Web Services provide powerful yet simple, highly abstracted, imaginative, telecommunications functions that application developers and the IT community can both quickly comprehend and use to generate new, innovative applications.

The Open Service Access (OSA) specifications define an architecture that enables application developers to make use of network functionality through an open standardized interface, i.e. the Parlay/OSA APIs.

2 References

For the purposes of this Technical Report (TR), the following references apply:

[1] ETSI TR 121 905: "Universal Mobile Telecommunications System (UMTS); Vocabulary for 3GPP Specifications (3GPP TR 21.905)".

[2] W3C Recommendation (2 May 2001): "XML Schema Part 2: Datatypes".

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

[3] ETSI TR 102 397-1: "Open Service Access (OSA); Mapping of Parlay X Web Services to Parlay/OSA APIs; Part 1: Common Mapping".

[4] ETSI ES 202 391-13: "Open Service Access (OSA); Parlay X Web Services; Part 13: Address List Management".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 102 397-1 [3] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TR 102 397-1 [3] apply.

4 Mapping description

The Call Handling capability can be implemented with Parlay/OSA Multi-Party Call Control and User Interaction.

It is applicable to ETSI OSA 1.x/2.x/3.x, Parlay/OSA 3.x/4.x/5.x and 3GPP Releases 4 to 6.

5 Sequence diagrams

5.1 Enabling call notifications

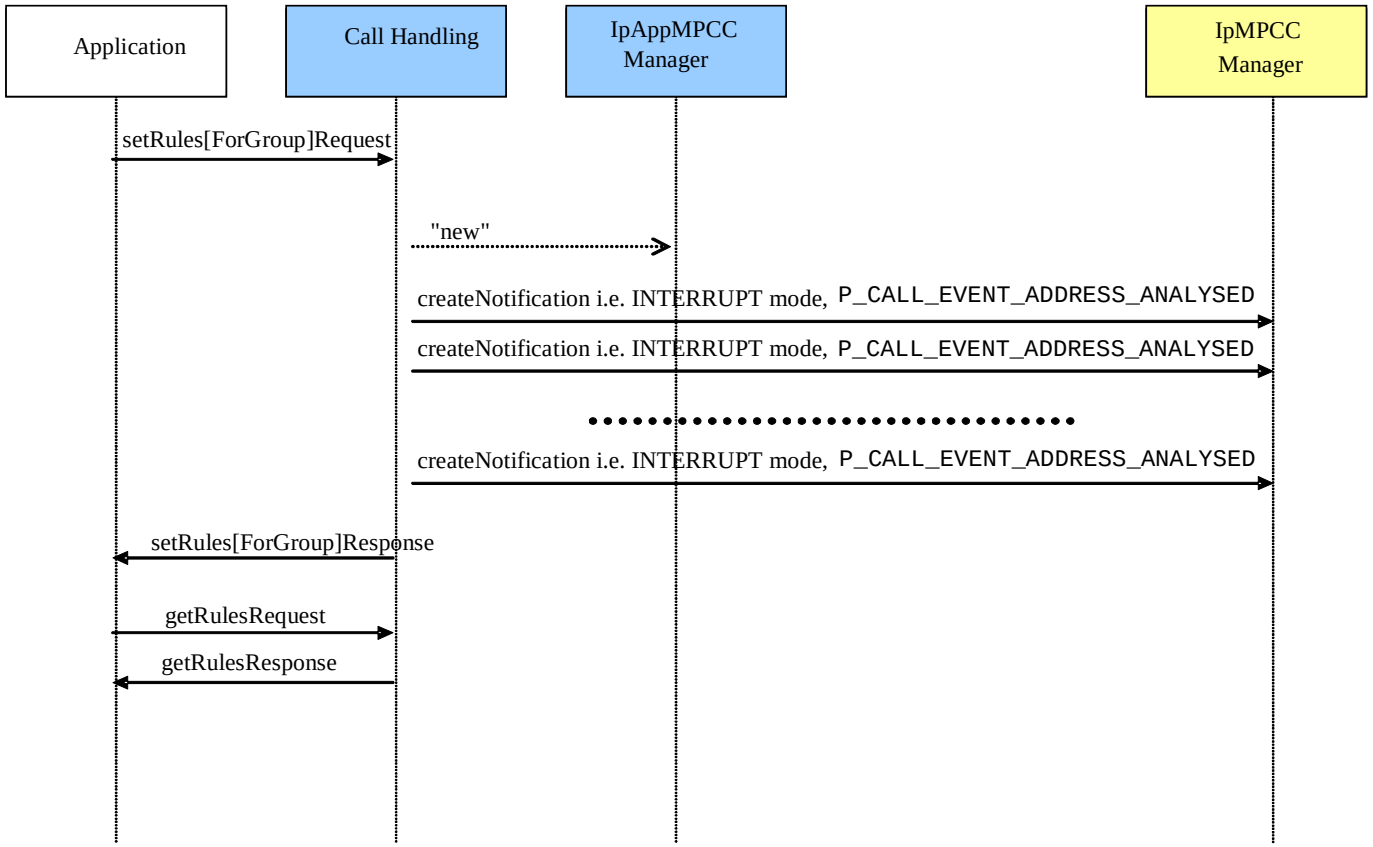


Figure 1

5.2 Disabling call notifications

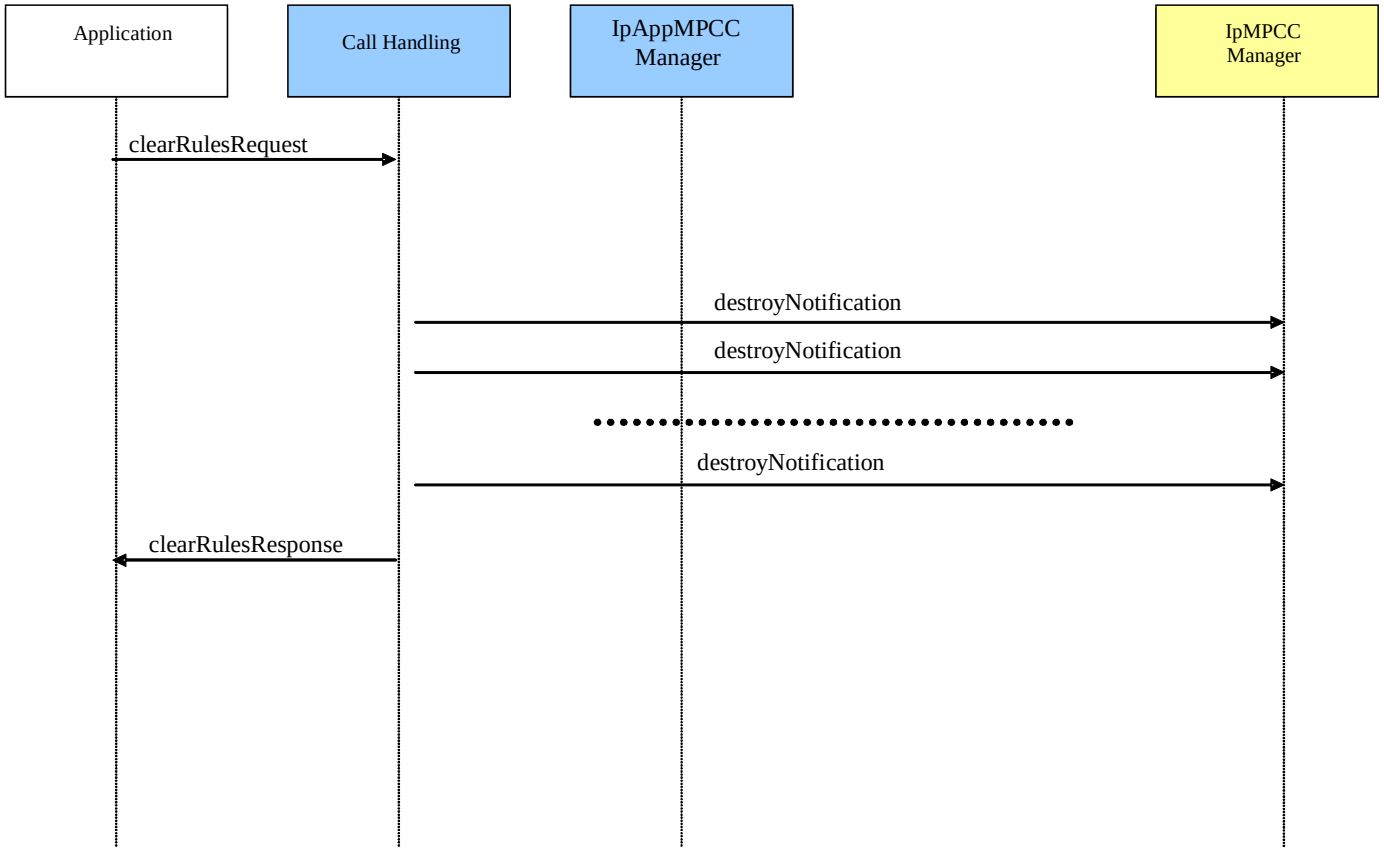


Figure 2

5.3 Processing a call: Route to original destination

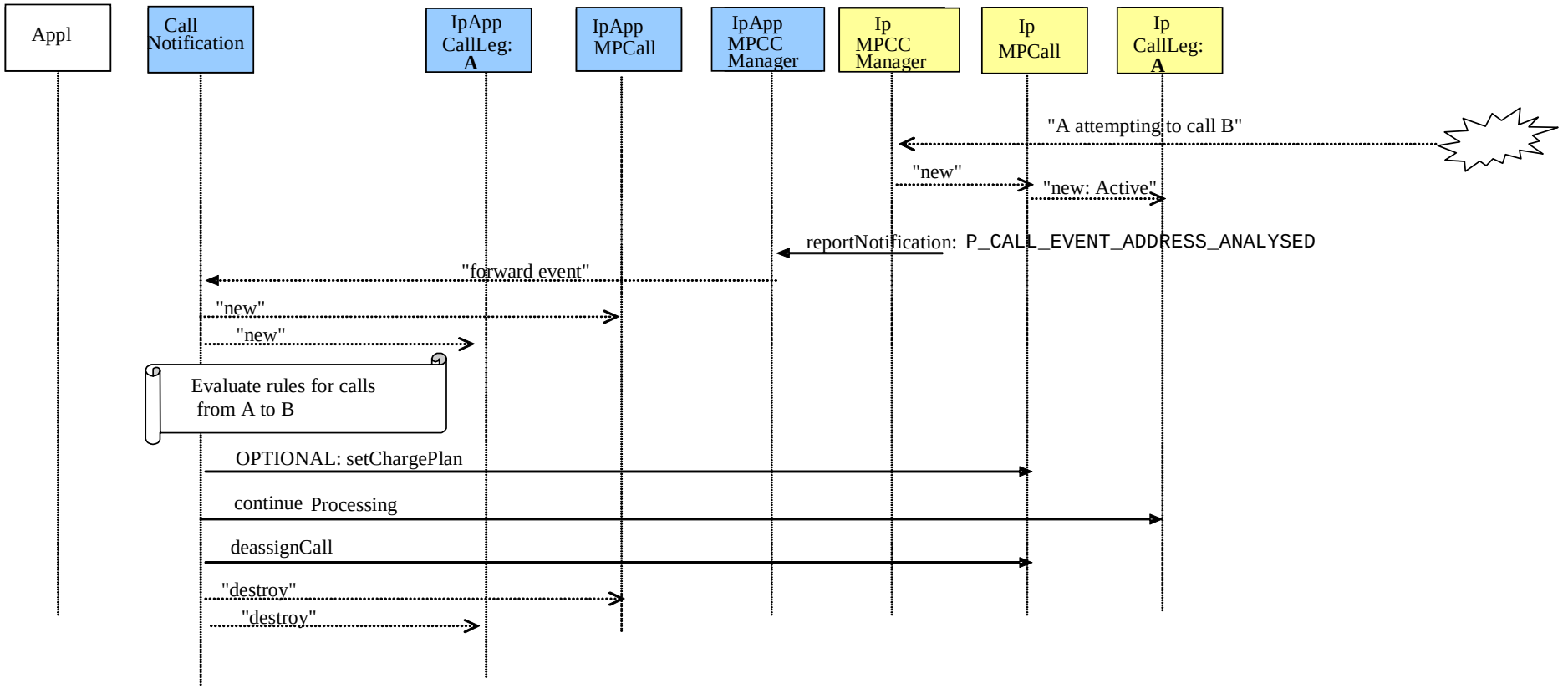


Figure 3

5.4 Processing a call: Perform user interaction & terminate

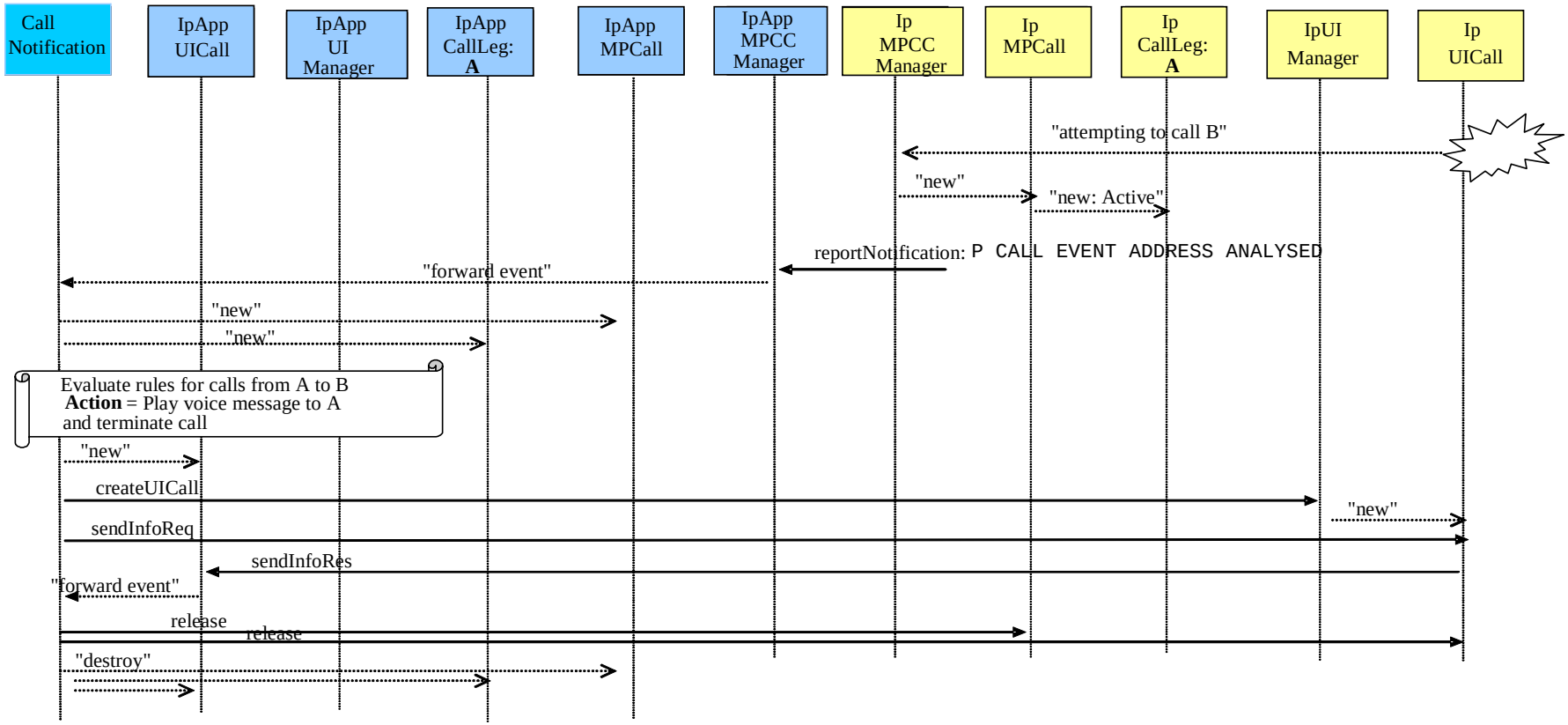


Figure 4

5.5 Processing a call: Forward to "Busy" destination, re-route to original destination

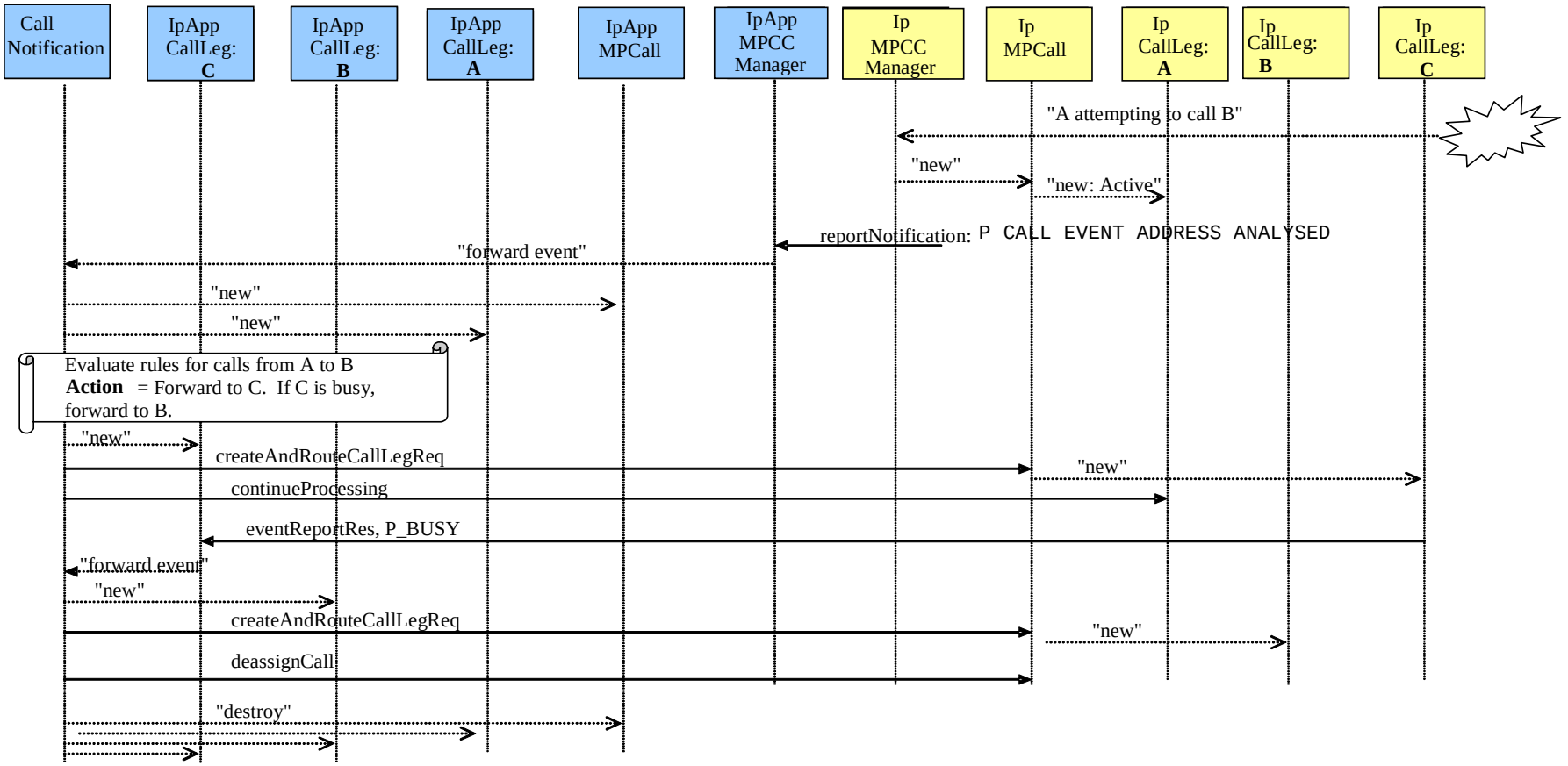


Figure 5

6 Detailed mapping information

6.1 Operations

The Call Handling web service operations are mapped to the Parlay/OSA APIs in two distinct areas:

- enabling and disabling of call notifications associated with originating call attempts, specifically the "address analyzed" trigger, which is discussed in clauses 6.1.1 through 6.1.3;
- sequential rule-based processing of originating call attempts, which is discussed (in order of precedence) in clauses 6.1.4 through 6.1.10.

6.1.1 setRules and setRulesForGroup

These operations set up (or replace as applicable) the set of rules associated with a destination address or addresses. Call notification is established with the Multi-Party Call Control service for the termination address for the receipt of a call attempt notification.

The sequence diagram in clause 5.1 illustrates the flow for these operations. They are mapped to the Parlay/OSA method: `IpMultiPartyCallControlManager.createNotification`.

6.1.1.1 Mapping to `IpMultiPartyCallControlManager.createNotification`

The `IpMultiPartyCallControlManager.createNotification` method is invoked with the following parameters.

Name	Type	Comment
appCallControl Manager	IpAppMultiPartyCall ControlManagerRef	Specifies the interface for receiving call-related event notifications associated with the criteria contained in the `notificationRequest`.
notification Request	TpCallNotification Request	Specifies event-related data, which is mapped from the parts of the **setRules[ForGroup]Request** message as described in clause 6.1.1.2.

The result from `IpMultiPartyCallControlManager.createNotification` is of type TpAssignmentID. It is used internally by the Call Handling web service to correlate the Parlay/OSA callbacks, e.g. `IpAppMultiPartyCallControlManager.reportNotification`, when a call-related event, which is associated with the criteria contained in the notificationRequest, is triggered in the network.

Parlay exceptions thrown by `IpMultiPartyCallControlManager.createNotification` are mapped to Parlay X exceptions as defined in clause 6.1.4.

6.1.1.2 Mapping from **setRules[ForGroup]Request** to `notificationRequest`

The elements of the `notificationRequest` data type are derived from the parts of the **setRules[ForGroup]Request** message as follows.

Name	Type	Comment
CallNotification Scope	TpCallNotification Scope	Specifies the destination address of the call, which is derived from the URI in the **address[es]** part of **setRules[ForGroup]Request**, as described in TR 102 397-1 [3].
CallEvents Requested	TpCallEventRequest Set	Defines a SINGLE element of a set, as follows: • `CallEventType = P_CALL_EVENT_ADDRESS_ANALYSED` • `AdditionalCallEventCriteria` = Null • `CallMonitorMode` = `P_CALL_MONITOR_MODE_INTERRUPT`.

6.1.2 getRules

This operation does not interact with any network elements, it returns the configured rules for an address.

The sequence diagram in clause 5.1 illustrates the flow for this operation.

6.1.3 clearRules

This operation disables the call notification from the Multi-Party Call Control service, and clears the related rules information stored by the service.

The sequence diagram in clause 5.2 illustrates the flow for these operations. They are mapped to the Parlay/OSA method: `IpMultiPartyCallControlManager.destroyNotification`.

6.1.3.1 Mapping to `IpMultiPartyCallControlManager.destroyNotification`

The `IpMultiPartyCallControlManager.destroyNotification` method is invoked with the following parameters.

Name	Type	Comment
assignmentID	TpAssignmentID	Specifies the assignment id returned after an earlier invocation of `IpMultiPartyCallControlManager.createNotification` method.

Parlay exceptions thrown by `IpMultiPartyCallControlManager.destroyNotification` are mapped to Parlay X exceptions as defined in clause 6.1.4.

6.1.4 CallHandlingRules.AcceptList Rule Processing

Call accepting determines if the call is accepted or rejected.

If the accept list associated with the called party (B) is null, then CallHandlingRules.AcceptList rule processing ends: processing continues with CallHandlingRules.BlockList Rule Processing.

If the calling party (A) is not a member of B's accept list, the call attempt by A is rejected: processing continues with clause 6.1.10 CallHandlingRules: Reject Call Attempt.

Otherwise the calling party (A) is a member of B's accept list, the call attempt by A is not rejected: processing continues with CallHandlingRules.ForwardList Rule Processing.

Rule processing is invoked by the Parlay/OSA method: `IpAppCallControlManager.reportNotification`, as illustrated in clauses 5.3 to 5.5.

6.1.4.1 Mapping from `IpAppMultiPartyCallControlManager.reportNotification`

The `IpAppMultiPartyCallControlManager.reportNotification` method is invoked with the following parameters.

Name	Type	Comment
callReference	TpMultiParty CallIdentifier	Specifies the reference to the call interface to which the notification relates.
callLegReferenceSet	TpCallLegIdentifierSet	Specifies the set of all call leg references associated with the call. Contains a single reference to the calling party (A) leg.
notificationInfo	TpCallNotificationInfo	Specifies event-related data, which is mapped to the Call Handling rule database as described in clause 6.1.4.2.
assignmentID	TpAssignmentID	Specifies the assignment id returned after an earlier invocation of `IpMultiPartyCallControlManager.createNotification` method, when the criteria associated with this call-related event were activated in the network, as described in clause 6.1.1.1. It is used internally by the Call Notification web service to correlate the Parlay/OSA callbacks.

The result from `IpAppMultiPartyCallControlManager.reportNotification` is of type `TpAppMultiPartyCallBack` (element=`AppMultiPartyCallAndCallLeg`), which specifies references to the application interfaces which implement the callback interfaces for the call and the calling party (A) leg.

6.1.4.2 Mapping from `TpCallNotificationInfo` to Call Handling Rule Database

The elements of the `TpCallNotificationInfo` data type are mapped to the Call Handling rule database as follows.

Name	Type	Comment

CallNotification ReportScope	TpCallNotification ReportScope	Specifies the called party address (B) and calling party address (A) of the call.
		• B's address is used as an index into the rule database to locate the **CallHandlingRules** structure for B. Mapped to the URI in the **address(es)** part of a **setRules(ForGroup)Request** message, as described in TR 102 397-1 [3].
		• A's address is mapped to a URI provided in the **CallHandlingRules** structure for B, as described in TR 102 397-1 [3]. Matched against the contents of one or more of the following lists:
		  • **CallHandlingRules.{AcceptList}**
		  • **CallHandlingRules.{BlockList}**
		  • **CallHandlingRules.{ForwardList.CallingAddress}.**

CallAppInfo	TpCallAppInfoSet	Not mapped.
CallEventInfo	TpCallEventInfo	Contains the event which is reported, which is mapped as described in the following table.

The elements of the `TpCallEventInfo` data type are mapped as follows.

Name	Type	Comment
CallEventType	TpCallEventType	Not mapped. This element has a value of `P_EVENT_GCCS_ADDRESS_ANALYSED_EVENT`.
AdditionalCallEvent Info	TpCallAdditionalEvent Info	For `P_CALL_EVENT_ADDRESS_ANALYSED` this element contains B's address, which is redundant here and ignored.
CallMonitorMode	TpCallMonitorMode	Not mapped. This element has a value of "`P_CALL_MONITOR_MODE_INTERRUPT`".
CallEventTime	TpDateAndTime	Not mapped.

6.1.5 CallHandlingRules.BlockList Rule Processing

Call blocking determines if the call is rejected.

If the block list associated with the called party (B) is null, the call attempt by A is not rejected: processing continues with CallHandlingRules.ForwardList Rule Processing.

If the calling party (A) is not a member of B's block list, the call attempt by A is not rejected: processing continues with CallHandlingRules.ForwardList Rule Processing.

Otherwise the calling party (A) is a member of B's block list, the call attempt by A is rejected: processing continues with clause 6.1.10 CallHandlingRules: Reject Call Attempt.

Rule processing is invoked by the Parlay/OSA method: `IpAppMultiPartyCallControlManager.reportNotification` as described in clauses 6.1.4.1 and 6.1.4.2, and as illustrated in clauses 5.3 to 5.5.

6.1.6 CallHandlingRules.ForwardList Rule Processing

Conditional call forwarding determines how the call attempt is forwarded, and possibly re-forwarded if the forwarded call does not complete.

If the (conditional) forward list associated with the called party (B) is null, the call attempt by A is not rejected: processing continues with CallHandlingRules.Forward Rule Processing.

If the calling party (A) is not a member of B's forward list - i.e. A ∉ **{ForwardList.ConditionalForward. CallingAddress}** - the call attempt by A is not rejected: processing continues with CallHandlingRules.Forward Rule Processing.

Otherwise the calling party (A) is a member of B's forward list – i.e. A ∈ **{ForwardList.ConditionalForward. CallingAddress}** - the call attempt by A is forwarded to C (**ForwardList.ConditionalForward.ForwardingAddress**). In this case, zero, one or both call-related event reports are requested, as follows:

- interrupt call processing for a "C is busy" event, if **ForwardList.ConditionalForward.OnBusyAddress** is non-null;
- interrupt call processing for a "no answer from C" event, if **ForwardList.ConditionalForward. OnNoAnswerAddress** is non-null.

If no call-related event reports are requested, then rule processing is completed.

If the "C is busy" call event is triggered, the call attempt by A is re-forwarded to D (**ForwardList.ConditionalForward.OnBusyAddress**). No call-related event reports are requested. Rule processing is completed.

If the "no answer from C" call event is triggered, the call attempt by A is re-forwarded to E (**ForwardList.ConditionalForward. OnNoAnswerAddress**). No call-related event reports are requested. Rule processing is completed.

Otherwise, no requested call events are triggered on the forwarding to C and rule processing is completed.

Rule processing is originally invoked by the Parlay/OSA method: `IpAppMultiPartyCallControlManager.reportNotification` as described in clauses 6.1.4.1 and 6.1.4.2. Conditional call forward rule processing maps to/from the following Parlay/OSA methods, as illustrated in clause 5.5:

- `IpMultiPartyCall.createAndRouteCallLegReq; OR`

  `{IpMultiPartyCall.createCallLeg, IpCallLeg.eventReportReq,IpCallLeg.routeReq}`

- `IpAppCallLeg.eventReportRes;`
- `IpAppMultiPartyCall.createAndRouteCallLegErr; OR`

  `IpAppCallLeg.routeErr;`

- `IpAppCallLeg.eventReportErr.`

6.1.6.1 Mapping to IpMultiPartyCall.createAndRouteCallLegReq

The IpMultiPartyCall.createAndRouteCallLegReq method is invoked with the following parameters.

Name	Type	Comment
callSessionID	TpSessionID	Not mapped: derived from the callReference parameter of IpAppMultiPartyCallControlManager.report Notification, as described in clause 6.1.4.1.
eventsRequested	TpCallEventRequest Set	Depending on the content of the Conditional Forward list entry for calling party A, as discussed in clause 6.1.6, this set contains at most one entry. If the entry exists, it has the following elements: • CallEventType = P_CALL_EVENT_TERMINATING_RELEASE • AdditionalCallEventCriteria.TerminatingReleaseCa useSet = {P_BUSY and/or P_NO_ANSWER} • CallMonitorMode = P_CALL_MONITOR_MODE_INTERRUPT.
targetAddress	TpAddress	Specifies the destination leg to which the call should be routed: i.e forwarded call party C, or re-forwarded call party D or E, as described in clause 6.1.6, and mapped as described in TR 102 397-1 [3].
originatingAddress	TpAddress	Not mapped: derived from the notificationInfo parameter of IpAppMultiPartyCallControlManager.report Notification.
appInfo	TpCallAppInfoSet	Not mapped: derived from the notificationInfo parameter of IpAppMultiPartyCallControlManager.report Notification.
appLegInterface	IpAppCallLegRef	Not mapped: [Specifies a reference to the application interface that implements the callback interface for the new call leg. Requested events will be reported by the eventReportRes() operation on this interface.]

The result from IpMultiPartyCall.createAndRouteCallLegReq is of type TpCallLegIdentifier and is not mapped to the Parlay X interface.

Parlay exceptions thrown by IpMultiPartyCall.createAndRouteCallLegReq are not mapped to Parlay X exceptions. Instead, processing continues with CallHandlingRules: Continue Existing Call Attempt.

An alternative to mapping to the IpMultiPartyCall.createAndRouteCallLegReq convenience method is a mapping to the following discrete method invocations

- *IpMultiPartyCall.createCallLeg;*
- *IpCallLeg.eventReportReq;*
- *IpCallLeg.routeReq.*

6.1.6.1.1 Alternative Mapping to IpMultiPartyCall.createCallLeg

The IpMultiPartyCall.createCallLeg method is invoked with the following parameters.

Name	*Type*	*Comment*
callSessionID	*TpSessionID*	*Not mapped: derived from the callReference parameter of IpAppMultiPartyCallControlManager.reportNotificatio n, as described in* clause 6.1.4.1.
appCallLeg	*IpAppCallLegRef*	*Not mapped: [Specifies a reference to the application interface that implements the callback interface for the new call leg. Requested events will be reported by the eventReportRes() operation on this interface.]*

The result from IpMultiPartyCall.createCallLeg is of type TpCallLegIdentifier and is not mapped to the Parlay X interface.

Parlay exceptions thrown by IpMultiPartyCall.createCallLeg are not mapped to Parlay X exceptions. Instead, processing continues with CallHandlingRules: Continue Existing Call Attempt.

6.1.6.1.2 Alternative Mapping to IpCallLeg.eventReportReq

The IpCallLeg.eventReportReq method is invoked with the following parameters.

Name	Type	Comment
callLegSessionID	*TpSessionID*	*Not mapped: the result returned from the invocation of IpMultiPartyCall.createCallLeg, as described in clause 6.1.6.1.1.*
eventsRequested	*TpCallEventRequest Set*	*Depending on the content of the Conditional Forward list entry for calling party A, as discussed in clause 6.1.6, this set contains at most one entry. If the entry exists, it has the following elements:* • *CallEventType = P_CALL_EVENT_TERMINATING_RELEASE* • *AdditionalCallEventCriteria.TerminatingReleaseCauseSet = {P_BUSY and/or P_NO_ANSWER}* • *CallMonitorMode = P_CALL_MONITOR_MODE_INTERRUPT.*

Parlay exceptions thrown by IpCallLeg.eventReportReq are not mapped to Parlay X exceptions. Instead, processing continues with CallHandlingRules: Continue Existing Call Attempt.

6.1.6.1.3 Alternative Mapping to IpCallLeg.routeReq

The IpCallLeg.routeReq method is invoked with the following parameters.

Name	Type	Comment
callLegSessionID	*TpSessionID*	*Not mapped: the result returned from the invocation of IpMultiPartyCall.createCallLeg, as described in clause 6.1.6.1.1.*
targetAddress	*TpAddress*	*Specifies the destination leg to which the call should be routed: i.e forwarded call party C, or re-forwarded call party D or E, as described in clause 6.1.6, and mapped as described in TR 102 397-1 [3].*
originatingAddress	*TpAddress*	*Not mapped: derived from the notificationInfo parameter of IpAppMultiPartyCallControlManager.reportNotification.*
appInfo	*TpCallAppInfoSet*	*Not mapped: derived from the notificationInfo parameter of IpAppMultiPartyCallControlManager.reportNotification.*
connectionProperties	*TpCallLegConnection Properties*	*Not mapped. Specifies the properties of the connection: i.e. AttachMechanism = P_CALLLEG_ATTACH_IMPLICITLY.*

Parlay exceptions thrown by IpCallLeg.routeReq are not mapped to Parlay X exceptions. Instead, processing continues with CallHandlingRules: Continue Existing Call Attempt.

6.1.6.2 Mapping from IpAppCallLeg.eventReportRes

The IpAppCallLeg.eventReportRes callback method is invoked with the following parameters.

Name	Type	Comment
callLegSessionID	TpSessionID	Not mapped. [The value provide in the result from IpMultiPartyCall.create(andRoute)CallLeg(Req)].
eventInfo	TpCallEventInfo	Contains the following elements: • CallEventType = P_CALL_EVENT_TERMINATING_RELEASE • AdditionalCallEventInfo.TerminatingReleaseCause = either P_BUSY or P_NO_ANSWER • CallMonitorMode = P_CALL_MONITOR_MODE_INTERRUPT • CallEventTime. The present document specifies the reason the call could not be forwarded to call party C (reference the discussion in clause 6.1.6). Results in call being re-forwarded to call party D or E.

6.1.6.3 Mapping from `IpAppMultiPartyCall.createAndRouteCallLegErr`

The `IpAppMultiPartyCall.createAndRouteCallLegErr` callback method is invoked with the following parameters.

Name	Type	Comment
callSessionID	TpSessionID	Not mapped: the value provided in the `callReference` parameter of `IpAppMultiPartyCallControlManager.reportNotification`, as described in clause 6.1.4.1.
callLegReference	TpCallLegIdentifier	Not mapped. Specifies the reference to the CallLeg interface that was created and routed unsuccessfully.
errorIndication	TpCallError	Not mapped. Specifies the error which led to the original request failing.

Since conditional call forward rule processing is unsuccessful, the call attempt is allowed to continue to the original called party B: i.e. processing continues with CallHandlingRules: Continue Existing Call Attempt.

An alternative to mapping from the `IpMultiPartyCall.createAndRouteCallLegErr` convenience method is a mapping from the discrete method `IpAppCallLeg.routeErr`.

6.1.6.3.1 Alternative Mapping from `IpAppCallLeg.routeErr`

The `IpAppCallLeg.routeErr` callback method is invoked with the following parameters.

Name	*Type*	*Comment*
callLegSessionID	*TpSessionID*	*Not mapped. [The value provide in the result from `IpMultiPartyCall.createCallLeg`].*
errorIndication	*TpCallError*	*Not mapped. Specifies the error which led to the original request failing.*

Since conditional call forward rule processing is unsuccessful, the call attempt is allowed to continue to the original called party B: i.e. processing continues with CallHandlingRules: Continue Existing Call Attempt.

6.1.6.4 Mapping from `IpAppCallLeg.eventReportErr`

The `IpAppCallLeg.eventReportErr` callback method is invoked with the following parameters.

Name	Type	Comment
callLegSessionID	TpSessionID	Not mapped. [The value provide in the result from `IpMultiPartyCall.create(andRoute)CallLeg(Req)`].
errorIndication	TpCallError	Not mapped. Specifies the error which led to the original request failing.

Since conditional call forward rule processing is unsuccessful, the call attempt is allowed to continue to the original called party B: i.e. processing continues with CallHandlingRules: Continue Existing Call Attempt.

6.1.7 CallHandlingRules.Forward Rule Processing

Unconditional call forwarding determines how the call attempt is forwarded, and possibly re-forwarded if the forwarded call does not complete.

If the (unconditional) forward list associated with the called party (B) is null, the call attempt by A is not rejected: processing continues with CallHandlingRules.VoiceInteractionContent Rule Processing.

Otherwise the call attempt by A is forwarded to C (**ForwardList.UnconditionalForward.ForwardingAddress**). In this case, zero, one or both call-related event reports are requested, as follows:

- interrupt call processing for a "C is busy" event, if **ForwardList.UnconditionalForward.OnBusyAddress** is non-null;
- interrupt call processing for a "no answer from C" event, if **ForwardList.UnconditionalForward. OnNoAnswerAddress** is non-null.

If no call-related event reports are requested, then rule processing is completed.

If the "C is busy" call event is triggered, the call attempt by A is re-forwarded to D (**ForwardList.UnconditionalForward.OnBusyAddress**). No call-related event reports are requested. Rule processing is completed.

If the "no answer from C" call event is triggered, the call attempt by A is re-forwarded to E (**ForwardList.UnconditionalForward.OnNoAnswerAddress**). No call-related event reports are requested. Rule processing is completed.

Otherwise, no requested call events are triggered on the forwarding to C and rule processing is completed.

Rule processing is originally invoked by the Parlay/OSA method: `IpAppMultiPartyCallControlManager.reportNotification` as described in clauses 6.1.4.1 and 6.1.4.2. Unconditional call forward rule processing maps to/from the following Parlay/OSA methods, as illustrated in clause 5.5:

- `IpMultiPartyCall.createAndRouteCallLegReq;`
- `IpAppCallLeg.eventReportRes;`
- `IpAppMultiPartyCall.createAndRouteCallLegErr;`
- `IpAppCallLeg.eventReportErr.`

The mapping of unconditional call forward rule processing to these methods is identical to the mapping of conditional call forward rule processing described in clauses 6.1.6.1 through 6.1.6.4, except that all references to clause 6.1.6 should be replaced with references to clause 6.1.7.

6.1.8 CallHandlingRules.VoiceInteractionContent Rule Processing

If there is no voice interaction content specified in the call handling rule database for called party B, processing continues with CallHandlingRules: Continue Existing Call Attempt.

Otherwise, the call is handled by a voice system, which handles all further processing of the call. Rule processing completes when the call is handed off. If call hand-off is unsuccessful, then processing continues with CallHandlingRules: Continue Existing Call Attempt.

Rule processing is originally invoked by the Parlay/OSA method: `IpAppMultiPartyCallControlManager.reportNotification` as described in clauses 6.1.4.1 and 6.1.4.2. Voice interaction content rule processing maps to/from the following Parlay/OSA methods, as illustrated in clause 5.4:

- `IpUIManager.createUICall;`
- `IpUICall.sendInfoReq;`
- `IpAppUICall.sendInfoRes;`
- `IpAppUICall.sendInfoErr.`

6.1.8.1 Mapping to `IpUIManager.createUICall`

The `IpUIManager.createUICall` method is invoked with the following parameters.

Name	Type	Comment
appUI	IpAppUICallRef	Not mapped: reference to callback (internal).
uiTargetObject	TpUITargetObject	Not mapped. [The value of the `callReference` parameter of `IpAppMultiPartyCallControlManager.reportNotification`, as described in clause 6.1.4.1].

The result from `IpUIManager.createUICall` is of type `TpUICallIdentifier` and is used internally to correlate the callbacks.

Parlay exceptions thrown by `IpUIManager.createUICall` are not mapped to Parlay X exceptions. Instead, processing continues with CallHandlingRules: Continue Existing Call Attempt.

6.1.8.2 Mapping to `IpUICall.sendInfoReq`

The `IpUICall.sendInfoReq` method is invoked with the following parameters.

Name	Type	Comment
userInteraction SessionID	TpSessionID	Not mapped: reference to callback (internal). [The value contained in the `TpUICallIdentifier` parameter returned by `IpUIManager.createUICall`].
info	TpUIInfo	The mapping from **VoiceInteraction.TextInfo** is described in clause 6.1.8.3 The mapping from **VoiceInteraction.VoiceXml** is described in clause 6.1.8.4 The mapping from **VoiceInteraction.Audio** is described in clause 6.1.8.5.
language	TpLanguage	
variableInfo	TpUIVariableInfoSet	
repeatIndicator	TpInt32	Not mapped.
response Requested	TpUIResponse Request	Not mapped. Set to `P_UI_FINAL_REQUEST`, i.e. no callback methods (`IpAppUICall.sendInfoRes/Err`) will be invoked.

The result from `IpUICall.sendInfoReq` is of type `TpAssignmentID` and is ignored.

Parlay exceptions thrown by `IpUICall.sendInfoReq` are not mapped to Parlay X exceptions. Instead, processing continues with CallHandlingRules: Continue Existing Call Attempt.

6.1.8.3 Mapping of VoiceInteraction.TextInfo

The **VoiceInteraction.TextInfo.Text** element is of type **xsd:string** and represents the text to process and play through a Text-To-Speech engine. It is mapped to the `info` and `variableInfo` parameters as follows:

- For ETSI OSA 1.x, Parlay/OSA 3.x and 3GPP Release 4.x and subsequent releases, the **Text** element is mapped to `InfoData` (`info.P_UI_INFO_DATA`), which defines the data to be sent to an end-user's terminal. The data is free-format and the encoding is depending on the resources being used.
- The Call Notification web service needs to indicate that text-to-speech processing is required from a network resource. Options for indicating this are vendor-specific.
- One option is to include an indicator in the `InfoData` parameter: e.g. by prefixing the value of the **Text** element.
- Another option is to use the v`ariableInfo` parameter: e.g. the `VariablePartInteger` or `VariablePartAddress` element.
- For ETSI OSA 3.x, Parlay/OSA 5.x and 3GPP Release 6.x, an alternative mapping of the **Text** element is to `InfoSynthData` (`info. P_UI_INFO_SYNTHESIS`), which describes the content and how the speech synthesis will be done. Specifically **VoiceInteraction.TextInfo.Text** is mapped to the `InfoSynthData.TextData` field. There is no mapping to the other fields of `InfoSynthData` that define how the synthesis should be done; these fields are provisioned by the vendor.

The **VoiceInteraction.TextInfo.Language** element is of type **xsd:string** and is mapped to the `language` parameter.

6.1.8.4 Mapping of VoiceInteraction.VoiceXml

The **VoiceInteraction.VoiceXml** element is of type **xsd:anyURI** and represents the location of VoiceXML to be processed by a VoiceXML browser. It is mapped to the `info` and `variableInfo` parameters as follows:

- For ETSI OSA 1.x, Parlay/OSA 3.x and 3GPP Release 4.x and subsequent releases, the **VoiceXml** element is mapped to `InfoAddress` (`info.P_UI_INFO_ADDRESS`), which defines the URL of the stream to be sent to an end-user's terminal.

NOTE: In later releases of the API, the scope of the `InfoAddress` parameter is expanded to represent the URL of a voice application script or stream to be either sent to an end-user's terminal or invoked in the network in order to carry out the interaction dialogue. However an alternative parameter mapping is also available in later API releases, as described below.

- The Call Notification web service needs to indicate that VoiceXML browser processing is required from a network resource. Options for indicating this are vendor-specific.
- One option is to provide an indicator in the v`ariableInfo` parameter: e.g. the `VariablePartInteger` or `VariablePartAddress` element.
- For ETSI OSA 3.x, Parlay/OSA 5.x and 3GPP Release 6.x, an alternative mapping of the **VoiceXml** element is to `InfoVXMLData` (`info. P_UI_INFO_VXML`), which defines the TpString that describes the VXML (Voice XML) page that is sent to the server for execution and interaction with the end-user. (See http://www.w3.org/TR/2000/NOTE-voicexml-20000505/ for more information.)

There is no mapping from the **VoiceInteraction.VoiceXml** element to the `language` parameter.

6.1.8.5 Mapping of VoiceInteraction.Audio

The **VoiceInteraction.Audio** element is of type **xsd:anyURI** and represents the location of audio content (WAV or MP3 file) to be played by an audio processor. It is mapped to the `info` and `variableInfo` parameters as follows:

- For ETSI OSA 1.x, Parlay/OSA 3.x and 3GPP Release 4.x and subsequent releases, the **Audio** element is mapped to `InfoAddress` (`info.P_UI_INFO_ADDRESS`), which defines the URL of the stream to be sent to an end-user's terminal.

NOTE: In later releases of the API, the scope of the `InfoAddress` parameter is expanded to represent the URL of a stream to be either sent to an end-user's terminal or invoked in the network in order to carry out the interaction dialogue. However an alternative parameter mapping is also available in later API releases, as described below.

- The Call Notification web service needs to indicate that audio processing is required from a network resource. Options for indicating this are vendor-specific.
- One option is to provide an indicator in the v`ariableInfo` parameter: e.g. the `VariablePartInteger` or `VariablePartAddress` element.
- For ETSI OSA 2.x, Parlay/OSA 4.x and 3GPP Release 5.x and subsequent releases, an alternative mapping of the **Audio** element is to `InfoWaveData` (`info.P_UI_INFO_WAVE`) or `InfoAuData` (`info.P_UI_INFO_AU`), which defines the WAVE or AU data to be sent to an end-user's terminal. Both these elements are of type `TpOctetSet` and should contain the URL value of the **Audio** element. If this is not possible, or if other audio formats are required (e.g. MP3 or others, as specified in the **AudioFormatsSupported** service policy), then the v`ariableInfo` parameter can also be used.

There is no mapping from the **VoiceInteraction.Audio** element to the `language` parameter.

6.1.9 CallHandlingRules: Continue Existing Call Attempt

If there are no rules specified in the call handling rule database for called party B, or an error occurs when implementing a rule action, then the call attempt is allowed to continue to the original called party B and rule processing is completed

Rule processing is originally invoked by the Parlay/OSA method: `IpAppMultiPartyCallControlManager.reportNotification` as described in clauses 6.1.4.1 and 6.1.4.2. The default action, continue existing call attempt, maps to the Parlay/OSA methods, `IpCallLeg.continueProcessing`, as illustrated in clause 5.3.

6.1.9.1 Mapping to `IpCallLeg.continueProcessing`

The `IpCallLeg.continueProcessing` method is invoked with the following parameters.

`callLegSessionID : in TpSessionID`

Specifies the call leg session ID of the call leg.

Name	Type	Comment
callLegSessionID	TpSessionID	Not mapped: derived from the `callLegReferenceSet` parameter of `IpAppMultiPartyCallControlManager.reportNotification`, as described in clause 6.1.4.1.

Parlay exceptions thrown by `IpCallLeg.continueProcessing` are not mapped to Parlay X exceptions.

6.1.10 CallHandlingRules: Reject Call Attempt

The call attempt from A to B may be rejected as a result of call acceptance or call blocking rule processing, as described in clauses 6.1.4 and 6.1.5, respectively, and rule processing is completed.

Reject call attempt processing maps to the Parlay/OSA `IpMultiPartyCall.release` method.

6.1.10.1 Mapping to `IpMultiPartyCall.release`

The `IpMultiPartyCall.release` method is invoked with the following parameters.

Name	Type	Comment
callSessionID	TpSessionID	Not mapped: derived from the `callReference` parameter of `IpAppCallControlManager.callEventNotify`, as described in clause 6.1.4.1.
cause	TpReleaseCause	Not mapped. Value should indicate application-directed termination of the call attampt.

Parlay exceptions thrown by `IpMultiPartyCall.release` are not mapped to Parlay X exceptions.

6.2 Exceptions

For the present document document, the mapping of Parlay/OSA API method exceptions to Parlay X Web Service exceptions is common and defined in TR 102 397-1 [3]. There are no service-specific exception mappings.

7 Additional notes

No additional notes.

History

Document history		
V1.1.1	August 2005	Publication