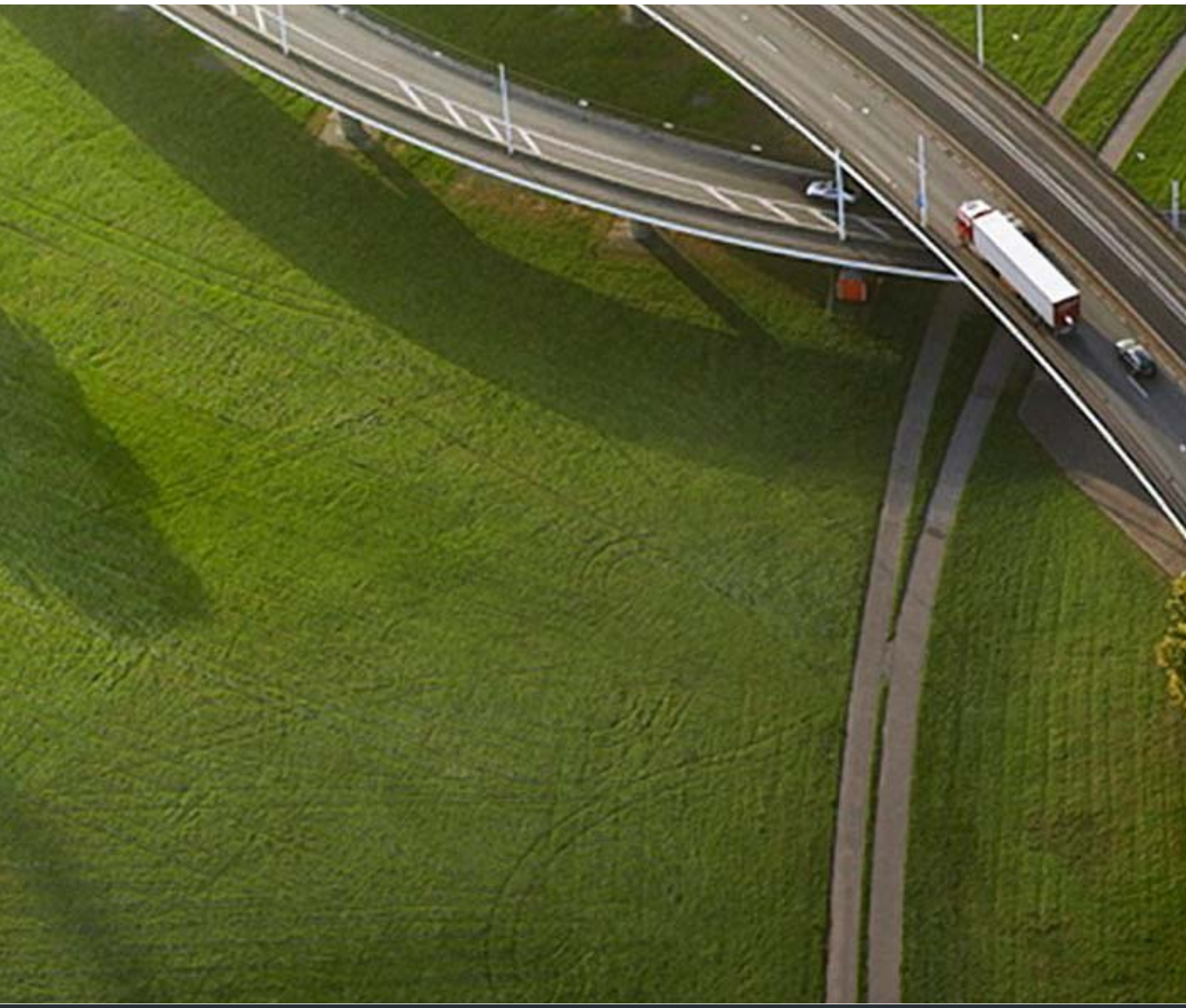
UNIFY Harmonize your enterprise



OpenScape 4000 CSTA V7
Connectivity Adapter - CSTA III, Part 2, Version 4.1

Developer's Guide

A31003-G9310-I200-1-76D1

Our Quality and Environmental Management Systems are implemented according to the requirements of the ISO9001 and ISO14001 standards and are certified by an external certification company.

Reference No.: A31003-G9310-I200-1-76D1

unify Harmonize your enterprise

unify.com

OpenScape 4000 CSTA V7

Connectivity Adapter – CSTA III, Part 2

Developer's Guide

Brief Overview

Detailed Table of Content

Content

Content

History of Change

Version	Date	Changes	Author
0.1	2001-12-18	initial draft	
0.2	2002-01-09	Layout adaption by documentation center	Janowicz
0.3	2002-02-08	release version	Kerndler
1.0	2002-02-19	revised	M. Bardehle
1.1	2002-03-14	minor changes by development	A. Horvath
1.2	2002-09-03	Presentation restricted concept	L. Czeh
1.3	2003-06-01	No functional, only formal (e.g. product version number) changes	L. Czeh
1.4	2004-02-23	New chapter: 5.19.8.2: Basic Internal Call with presentation restricted devices (CSTA3)	P. Hegedüs
3.0	2004-05-14	DigitsDialled event in all affected scenarios Call Forwarding scenarios	Zs.Ronkay
3.1	2004-12-13	New callscenario: 5.11.2. - Deflect Call service - deflect call from RCG Call Linkage Data in call scenarios 5.11.1. and 5.14.4.	P. Hegedüs
3.2	2004-12-21	Modifications in 5.12.1. Hold Call callscenario: different event cause and permitted services in case of keysets.	P. Hegedüs
3.3	2010-09	SIP and new name of product	K. Hideghethy
3.4	2012-01	Siemens name replacement	K. Hideghethy
4.0	2013-10	Rebranding to Unify	G. Foeldi
4.1	2014.04	V7 features ECMA conform handling of leaving multiple alerting scenarios (provide cause “multiAlert” for ConnectionCleared when group device leaves the call	Nagy, Andrea

The change bars in this document marks changes in the contents compared to the version: not applicable

5 Call Scenarios

5.1 Scope

This chapter contains the most common call scenarios of the OpenScape 4000 in OpenScape 4000 CSTA V1.
A call scenario is the series of steps that make up a telephony activity. A call scenario describes the actions occurring among all parties involved in a call, in sequence.

Each scenario includes a textual description and an illustration. Illustrations use the same key as described within ECMA-269. For each scenario, message sequences are listed for all device type monitored devices. All devices have device type monitors set with no events masked. The columns in each scenario represent the following:

- The Activity column includes a brief description of the telephony activity. The activity can either be initiated by a service invocation or manually.
- The Monitored Device(s) columns list events generated for the specified device type monitor or a service request and service response.
- The Comments column describes additional information on the activity.

The monitorCrossRefID parameter in events is not shown.

DeviceIDs are illustrated by Dn and ConnectionIDs in the from DnCn.

Table 5-1 Monitorable Devices

DeviceID	Description
Dn	Digital Telepone unless otherwise stated. (Attendant Console , Analog Telephone)
Rn	Route Control Group (RCG)
Gn	General Attendant (GA)
Hn	Hunt Group (HG)
Nn	Network Interface Device (NID)
An	Attendant Console

All Device IDs are within the same switching sub-domain unless otherwise indicated or stated. Any exception comments are made in the final column Comments.

We followed ECMA TR/82 as much as possible to make it easier for the application developer to compare the implementation of OpenScape 4000 with the ECMA directive. The document concentrates on the chosen CSTA implementation options and the differences from the ECMA directive.

Overview of the main sections:

The first 14 sections have the following structure:

Section 5.4, “Call Origination Scenarios”

Section 5.5, “Answering Call Scenarios”

Section 5.6, “Connection Termination Scenarios”

Section 5.7, “External outgoing calls”

Section 5.8, “External incoming calls”

Section 5.9, “Forwarding Call Scenarios”

Section 5.10, “Multiple Forwarding Scenarios”

Section 5.11, “Call Movement Scenarios”

Section 5.12, “Hold/Retrieving Scenarios”

Section 5.13, “Consultation Call Scenarios”

Section 5.14, “Transfer Call Scenarios”

Section 5.15, “Conference Call Scenarios”

Section 5.16, “Call Completion Scenarios”

Section 5.17, “Distribution Call Scenarios”

The ECMA-269 standard gives relative freedom to the implementation of recalls .The below section describes the interpretation of the OpenScape 4000.

Section 5.18, “Recall Scenarios”

The last section describes OpenScape 4000 features, that are either not described by ECMA 269 or they are not CSTA III standard compliant.

Section 5.19, “OpenScape Specific Features”

5.2 References

ECMA-269
Services for Computer Supported Telecommunications Applications (CSTA) Phase III, 4th edition (Dec 2011)

ECMA-285
Protocol for Computer Supported Telecommunications Applications (CSTA) Phase III, 2nd edition (Dec 2011)

ECMA TR/72
Glossary of definitions and terminology for Computer Supported Telecommunications Applications (CSTA) Phase III, 3rd edition (June 2009)

ECMA TR/82
Scenarios for Computer Supported Telecommunication Applications (CSTA) Phase III, (June 2009)

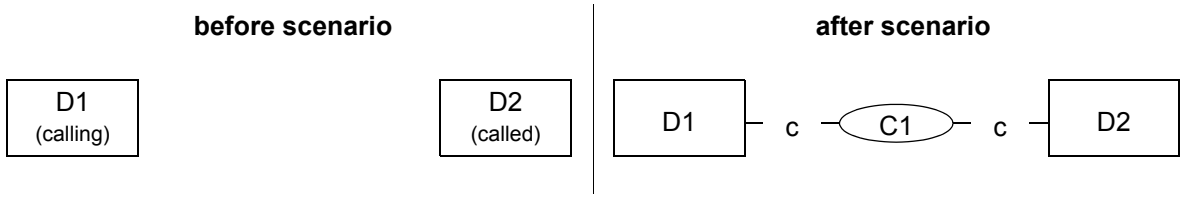
5.3 Definitions and Abbreviations

The definitions and abbreviations used in this Technical Report are defined in ECMA TR/72.

5.4 Call Origination Scenarios

5.4.1 Manually dialled call

This scenario illustrates a call originated through manual device activity.



Activity	Monitored Device D1		Monitored Device D2	Comments
1. D1 goes off-hook.	**Service Initiated**			
	• initiatedConnection	D1C1		
	• initiatingDevice	D1		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	ClearConn, DialDgt		
2. D1 completes dialling D2.	**Digits Dialled**			Number of D2 is: 1234
	• diallingConnection	D1C1		
	• diallingDevice	D1		
	• diallingSequence	“1234“		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	none		
	Originated			
	• originatedConnection	D1C1		
	• callingDevice	D1		
	• calledDevice	D2		
	• localConnectionInfo	connected		
	• cause	normal		
	• servicesPermitted	ClearConn		

Table 5-2 Manually dialled call (page 1 of 2)

Activity	Monitored Device D1		Monitored Device D2		Comments
3. D2 starts ringing.	**Delivered**		**Delivered**		
	• connection	D2C1	• connection	D2C1	
	• alertingDevice	D2	• alertingDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	alert	
	• cause	normal	• cause	normal	
	• servicesPermitted	CallBack, ClearConn, SendUserInfo	• servicesPermitted	AnswerCall, ClearConn, Deflect, SendUserInfo	
4. D2 answers the call.	**Established**		**Established**		
	• establishedConnection	D2C1	• establishedConnection	D2C1	
	• answeringDevice	D2	• answeringDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	connected	
	• cause	normal	• cause	normal	
	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	

Table 5-2 Manually dialled call (page 2 of 2)

Remark:

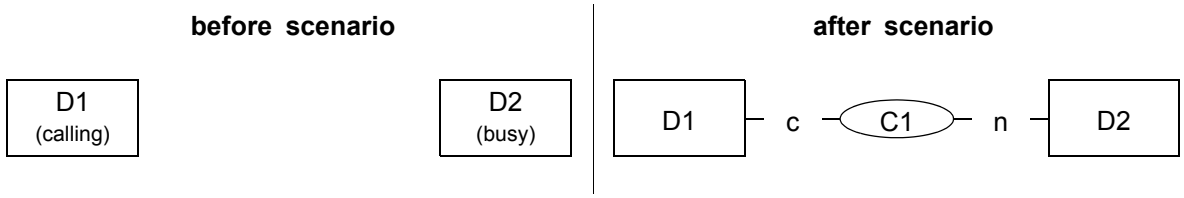
The complete dialled digits sequence is provided in the Originated event.

Digit Dialled events are never generated for manual activity.

A more specific event cause in the Service Initiated event cannot be provided.

5.4.2 Manually dialled call - called party is busy

This scenario illustrates a call scenario where a call is made to a busy party.



Activity	Monitored Device D1		Monitored Device D2		Comments
1. D1 goes off-hook.	**Service Initiated**				
	• initiatedConnection	D1C1			
	• initiatingDevice	D1			
	• localConnectionInfo	initiated			
	• cause	normal			
	• servicesPermitted	ClearConn, DialDgt			
2. D1 completes dialling D2.	**Digits Dialled**				Number of D2 is: 1234
	• diallingConnection	D1C1			
	• diallingDevice	D1			
	• diallingSequence	“1234“			
	• localConnectionInfo	initiated			
	• cause	normal			
	• servicesPermitted	none			
	Originated				
	• originatedConnection	D1C1			
	• callingDevice	D1			
	• calledDevice	D2			
	• lastRedirectionDevice	NS			
	• localConnectionInfo	connected			
	• cause	normal			
	• servicesPermitted	ClearConn			
3. D2 is busy. The call can not be completed. D1 hears busy tone.	**Failed**		**Failed**		This illustrates connection failures that report the Failed event for all devices involved with the call and that will provide a complete connectionID for the failed connection.
	• failedConnection	D2C1	• failedConnection	D2C1	
	• failingDevice	D2	• failingDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	fail	
	• cause	busy	• cause	busy	
	• servicesPermitted	ClearConn	• servicesPermitted	none	

Table 5-3 Unsuccessful basic call - called party is busy (page 1 of 2)

Activity	Monitored Device D1		Monitored Device D2		Comments
4. The busy connection is cleared immediately.	**Connection Cleared**		**Connection Cleared**		
	• droppedConnection	D2C1	• droppedConnection	D2C1	
	• releasingDevice	D2	• releasingDevice	D2	
	• localConnectionInfo	connected	• localConnectionInfo	null	
	• cause	normalClr	• cause	normalClr	
	• servicesPermitted	ClearConn	• servicesPermitted	none	

Table 5-3 Unsuccessful basic call - called party is busy (page 2 of 2)

Remark:

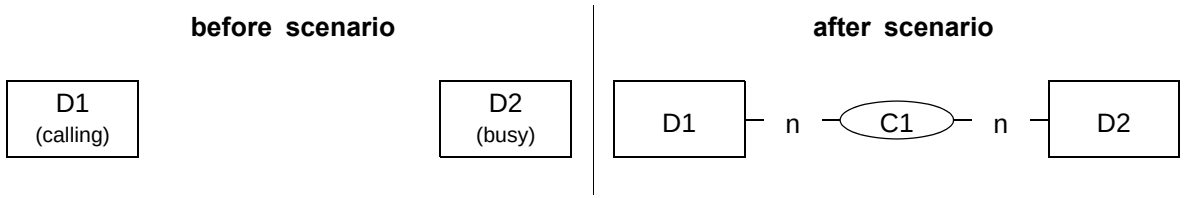
The protocol converter of the switching function will immediately send a Connection Cleared event after a connection goes into the busy failed state. This does not necessarily mean, that the connection physically goes to idle.

The complete dialled digits sequence is provided in the Originated event.

Digit Dialled events are never generated for manual activity.

5.4.3 Manually dialled call - called party is OOS (Out Of Service)

This scenario illustrates a call scenario where a call is made to an party, which is out of service.



Activity	Monitored Device D1		Monitored Device D2	Comments
1. **D1 goes offhook**	**Service Initiated**			
	• initiatedConnection	D1C1		
	• initiatingDevice	D1		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	ClearConn, DialDgt		

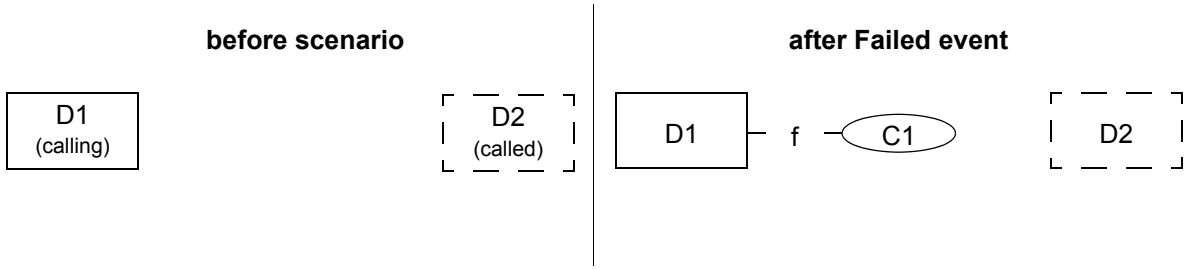
Activity	Monitored Device D1		Monitored Device D2		Comments
2. **D1 completes dialling D2's number**	**Digits Dialled**				Number of D2 is: 1234
	• diallingConnection	D1C1			
	• diallingDevice	D1			
	• diallingSequence	"1234"			
	• localConnectionInfo	initiated			
	• cause	normal			
	• servicesPermitted	none			
	•				
	Originated				
	• originatedConnection	D1C1			
	• callingDevice	D1			
	• calledDevice	D2			
	• lastRedirectionDevice				
	• localConnectinfo	connected			
	• cause	normal			
	• servicesPermitted	ClearConn			
3. **Called party D2 is Out of Service**	**Failed**		**Failed**		
	• failedConnection	D2C1	• failedConnection	D2C1	
	• failingDevice	D2	• failingDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	fail	
	• cause	destinationOutOfOrder	• cause	destinationOutOfOrder	
	• servicesPermitted	ClearConn	• servicesPermitted	none	
	Connection Cleared		**Connection Cleared**		
	• droppedConnection	D2C1	• doppedConnection	D2C1	
	• releasingDevice	D2	• releasingDevice	D2	
	• localConnectionInfo	connected	• localConnectionInfo	null	
	• cause	normalClr	• cause	normalClr	
	• servicesPermitted	ClearConn	• servicesPermitted	none	
4. **D1 goes onhook**	**Connection Cleared**				
	• droppedConnection	D1C1			
	• releasingDevice	D1			
	• localConnectionInfo	null			
	• cause	normalClr			
	• servicesPermitted	none			

Remark:

none

5.4.4 Manually dialled call - dialled number is invalid

This scenario illustrates a manually dialled call to an invalid destination. Device D2 is actually an invalid number.



Activity	Monitored Device D1		Monitored Device D2	Comments
1. Device D1 goes offhook.	**Service Initiated**			
	• initiatedConnection	D1C1		
	• initiatingDevice	D1		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	ClearConn, DialDigits		

Table 5-4 Dialled number invaild (page 1 of 2)

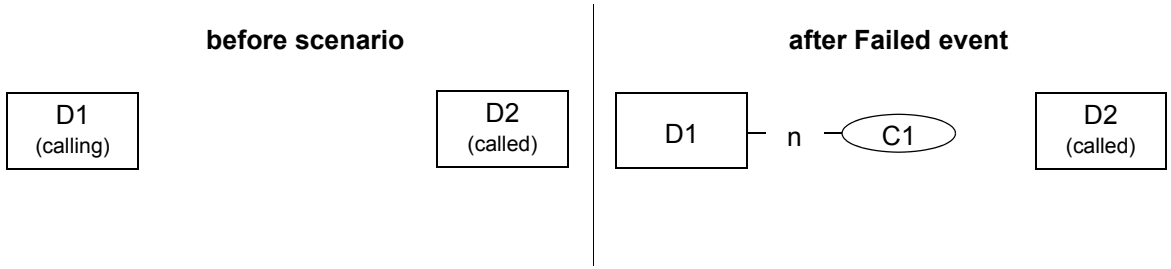
Activity	Monitored Device D1		Monitored Device D2	Comments
2. Since D1 dialled an invalid number, it becomes blocked.	**Digits Dialled**			D1 dials 9999- it is an invalid number
	• diallingConnection	D1C1		
	• diallingDevice	D1		
	• diallingSequence	“9999“		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	none		
	Failed			The switching function does not provide the Originated event in this case. D1C1 immidiately becomes failed.
	• failedConnection	D1C1		
	• failingDevice	D1		
	• callingDevice	D1		
	• calledDevice	NK		
	• lastRedirectionDevice	NS		
	• localConnectionInfo	fail		
	• cause	normal		
	• servicesPermitted	ClearConn		
3. Device D1 clears its failed call.	**Connection Cleared**			
	• droppedConnection	D1C1		
	• releasingDevice	D1		
	• localConnectionInfo	null		
	• cause	normalClr		
	• servicesPermitted	none		

Table 5-4 Dialled number invaild (page 2 of 2)

Remark: None

5.4.5 Manually dialled call - incomplete dialling sequence, calling party goes onhook

This scenario illustrates a manually dialled incomplete call. A starts a call to B. Before dialling the whole number A goes onhook.

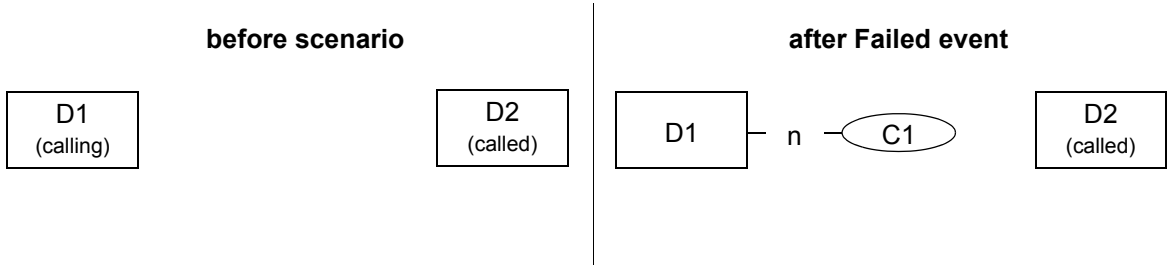


Activity	Monitored Device D1		Monitored Device D2	Comments
1. **D1 goes offhook**	**Service Initiated**			
	• initiatedConnection	D1C1		
	• initiatingDevice	D1		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	ClearConn, DialDgt		
2. **D1 does not complete dialling D2's number ("1234")**	**Digits Dialled**			Number of D2 is: 1234
	• diallingConnection	D1C1		
	• diallingDevice	D1		
	• diallingSequence	"123"		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	none		
3. **D1 goes onhook**	**Connection Cleared**			
	• droppedConnection	D1C1		
	• releasingDevice	D1		
	• localConnectionInfo	null		
	• cause	normal		
	• servicesPermitted	none		

5.4.6 Manually dialled call - incomplete dialling sequence, dialling has timed out

A starts a call to B. Dialling has timed out.

.

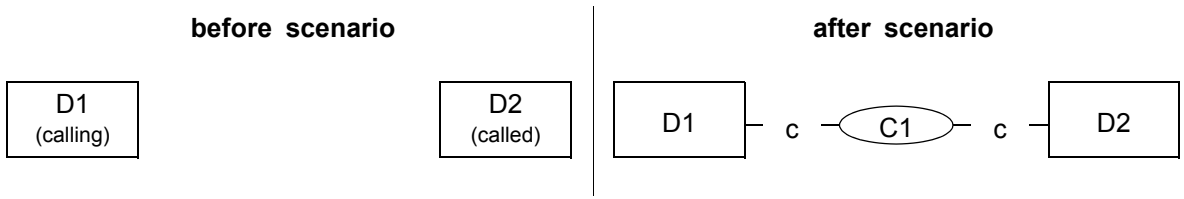


Activity	Monitored Device D1		Monitored Device D2	Comments
1. D1 goes offhook	**Service Initiated**			
	• initiatedConnection	D1C1		
	• initiatingDevice	D1		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	ClearConn, DialDgt		
2. D1 does not complete dialling D2's number ("1234")	**Digits Dialled**			Number of D2 is: 1234
	• diallingConnection	D1C1		
	• diallingDevice	D1		
	• diallingSequence	"123"		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	none		

Activity	Monitored Device D1		Monitored Device D2	Comments
3. dialling has timed out	**Failed**			
	• failedConnection	D1C1		
	• failingDevice	D1		
	• callingDevice	D1		
	• calledDevice	NK		
	• lastRedirectionDevice	NS		
	• localConnectionInfo	fail		
	• cause	normal		
	• servicesPermitted	ClearConn		
	Connection Cleared			
	• droppedConnection	D1C1		
	• releasingDevice	D1		
	• localConnectionInfo	null		
	• cause	normal		
	• servicesPermitted	none		

5.4.7 Make Call service

This scenario illustrates a successful Make Call from device D1 to device D2. In this scenario both devices are available and valid, device D1 is permitted to make the call and the call is answered by device D2.



Activity	Monitored Device D1		Monitored Device D2	Comments
1. Make call is invoked on D1.	**Make Call Request** • callingDeviceID • calledDirectoryNumber deviceID • autoAnswer	 D1 D2 prompt		The Make Call service specifies that device D1 should be prompted to go off-hook.
2. Acknowledgement	**Make Call Response** • connectionID	 D1C1		
3. Indication that the service initiated from this device	**Service Initiated** • initiatedConnection • initiatingDevice • localConnectionInfo • cause • servicesPermitted	 D1C1 D1 initiated MakeCall Answer, ClearConn, DialDgt, SendUserInfo		The MakeCall cause indicates that the device D1 is being prompted (via ringing, for example) to go off-hook.
Scenario proceeds as shown in "Manually dialled call" on page 5-4				

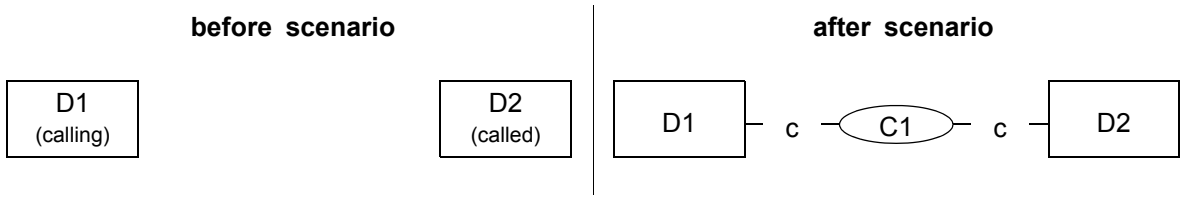
Table 5-5 Make call service

Remark:

None

5.4.8 Multi Stage dialling

This scenario illustrates the use of the Dial Digits service to complete dialling a call that was established via a Make Call service. In this scenario both devices are available and valid, device D1 is permitted to make the call and the call is answered by device D2 (3160).



Activity	Monitored Device D1		Monitored Device D2	Comments
1. Make call is invoked on D1.	**Make Call Request** • callingDeviceID • calledDirectoryNumber deviceID • autoAnswer	 D1 “31;” prompt		The Make Call service a partial dialling string that includes the first part of the number of D2 (“31”) and the partial dialling indicator (“;”).
2. Acknowledgement	**Make Call Response** • connectionID	 D1C1		
3. D1 goes off-hook. The event indicates that the service initiated from this device	**Service Initiated** • initiatedConnection • initiatingDevice • localConnectionInfo • cause • servicesPermitted	 D1C1 D1 initiated Make Call Answer, ClearConn, DialDgt, SendUserInfo		The MakeCall cause indicates that the device D1 is being prompted (via ringing, for example) to go off-hook.
4. The event indicates that partial dialling is used.	**Digits dialled** • diallingConnection • diallingDevice • diallingSequence • localConnectionInfo • cause • servicesPermitted	 D1C1 D1 “31;” initiated normal none		A “;” character indicates that there is an incomplete dialling string.
5. Dial Digits is invoked on D1.	**Dial Digits Request** • connectionToBeDialled • diallingSequence	 **D1C1** **“60”**		A “;” is not provided in the dialling string since there are no more digits to be dialled.
6. Acknowledgement	**Dial Digits Response**			
7. The dialling sequence is completed and D1 is connected in the call.	**Originated** • originatedConnection • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted	 D1C1 D1 D2 NS connected normal ClearConn, SendUserInfo		Note, that the last Digit Dialled event is missing. The switching function provides only the Originated event as an idicator of the finished dialling sequence. D2 is the called device. It contains the digits “3160” in this scenario.
Scenario proceeds as shown in “Manually dialled call” on page 5-4				

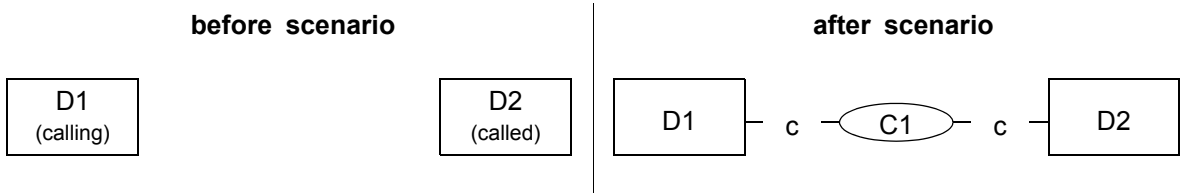
Table 5-6 Multi Stage Dialling

Remark:

None

5.4.9 Call offered to an application

This scenario illustrates a call offered to an application. Building up the call till Originated can happen either manually or via Make Call Service. Please refer to the previous tables. Offer is supported on monitored digital devices and requires special configuration. Providing Offered event to a calling device is optional.



Activity	Monitored Device D1		Monitored Device D2	Comments
1. D1 goes offhook	**Service Initiated**			
	• initiatedConnection	D1C1		
	• initiatingDevice	D1		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	ClearConn, DialDgt		

Table 5-7 Call offered (page 1 of 3)

Activity	Monitored Device D1		Monitored Device D2		Comments
2. D1 completes dialling D2	**Digits Dialled**				Number of D2 is: 1234
	• diallingConnection	D1C1			
	• diallingDevice	D1			
	• diallingSequence	“1234“			
	• localConnectionInfo	initiated			
	• cause	normal			
	• servicesPermitted	none			
	Originated				
	• originatedConnection	D1C1			
	• callingDevice	D1			
	• calledDevice	D2			
	• localConnectionInfo	connected			
	• cause	normal			
	• servicesPermitted	ClearConn			
3. Call is offered to D2	**Offered**		**Offered**		Providing Offered for calling side is optional
	• offeredConnection	D2C1	• offeredConnection	D2C1	
	• offeredDevice	D2	• offeredDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connectd	• localConnectionInfo	alerting	
	• cause	normal	• cause	normal	
	• servicesPermitted	ClearConn	• servicesPermitted	AcceptCall, ClearConn, Deflect	
4. Application accepts the call			**Accept Cal lRequest**		
			• callToBeAccepted	D2C1	
5. Acknowledged			**Accept Call Response**		
6. D2 starts ringing.	**Delivered**		**Delivered**		
	• connection	D2C1	• connection	D2C1	
	• alertingDevice	D2	• alertingDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	alert	
	• cause	normal	• cause	normal	
	• servicesPermitted	CallBack, ClearConn, SendUserInfo	• servicesPermitted	AnswerCall, ClearConn, Deflect, SendUserInfo	

Table 5-7 Call offered (page 2 of 3)

Activity	Monitored Device D1		Monitored Device D2		Comments
7. D2 answers the call.	**Established**		**Established**		
	• establishedConnection	D2C1	• establishedConnection	D2C1	
	• answeringDevice	D2	• answeringDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	connected	
	• cause	normal	• cause	normal	
	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	

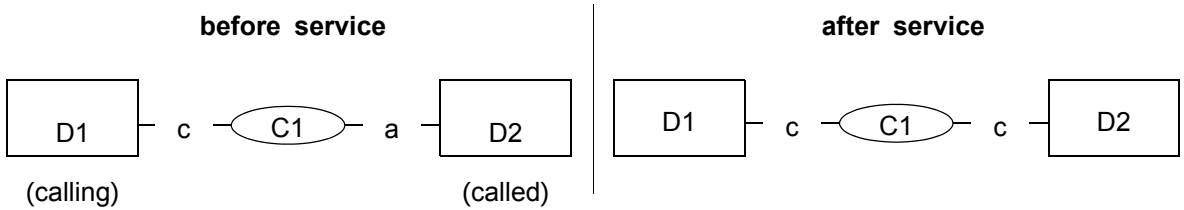
Table 5-7 Call offered (page 3 of 3)

Note: Providing Offered event also on the calling side is configurable.

5.5 Answering Call Scenarios

5.5.1 Succesful answer call

This clause illustrates how calls are answered by CSTA services.



See “Manually dialled call” on page 5-4 for the event flow to get into the “before service” state.

Activity	Monitored Device D1		Monitored Device D2		Comments
1. Answer call service is invoked on D2.			**Answer Call Request**		
			• call to answer call ID	D2C1	
2. Acknowledgement			**Answer Call Response**		
3. D2 answers the call.	**Established**		**Established**		
	• establishedConnection	D2C1	• establishedConnection	D2C1	
	• answeringDevice	D2	• answeringDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	connected	
	• cause	normal	• cause	normal	
	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	

Table 5-8 Answer Call service

Remark:

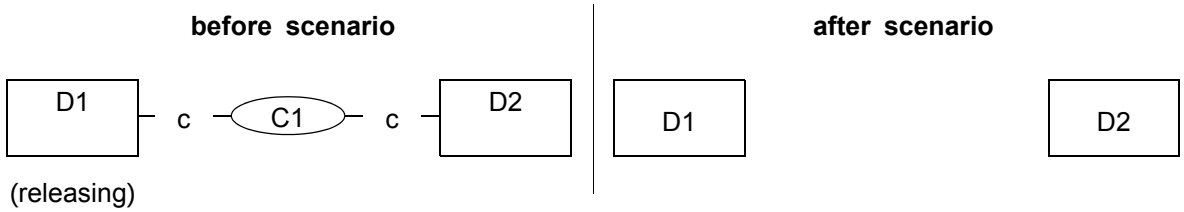
The manual case is similar to the described event flow.

5.6 Connection Termination Scenarios

5.6.1 Device disconnects from a call by on-hook

5.6.1.1 Non-SIP Device disconnects from a call by on-hook

In this scenario device D1 is manually put on-hook to release itself from the call. The remaining device goes blocked, until the device goes on-hook.



See “Succesful answer call” on page 5-19 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1		Monitored Device D2		Comments
1. D1 goes on-hook.	**Connection Cleared**		**Connection Cleared**		
	• droppedConnection	D1C1	• droppedConnection	D1C1	
	• releasingDevice	D1	• releasingDevice	D1	
	• localConnectionInfo	null	• localConnectionInfo	connected	
	• cause	normalClr	• cause	normalClr	
	• servicesPermitted	none	• servicesPermitted	ClearConn	

Table 5-9 Non-SIP Device disconnects from a call by on-hook(page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2		Comments
2. As a result of the "far end disconnect", the remaining connection D2C1 goes blocked.		**Failed**		
		• failedConnection	D2C1	
		• failingDevice	D2	
		• callingDevice	D1	
		• calledDevice	D2	
		• lastRedirectionDevice	NS	
		• localConnectionInfo	fail	
		• cause	blocked	
		• servicesPermitted	ClearConn	
3. The remaining device goes onhook.		**Connection Cleared**		
		• droppedConnection	D2C1	
		• releasingDevice	D2	
		• localConnectionInfo	null	
		• cause	normalClr	
		• servicesPermitted	none	

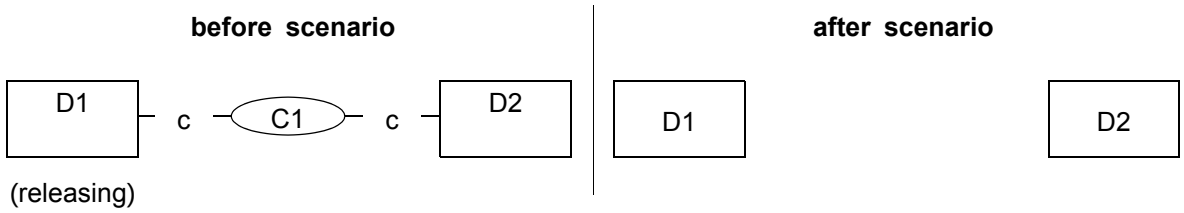
Table 5-9 Non-SIP Device disconnects from a call by on-hook(page 2 of 2)

Remark:

None

5.6.1.2 SIP Device disconnects from a call by on-hook

In this scenario device D1 (SIP) is manually put on-hook to release itself from the call. Both devices go blocked, until the devices go on-hook.



See "Succesful answer call" on page 5-19 for the event flow to get into the "before scenario" state.

Activity	Monitored Device D1		Monitored Device D2		Comments
1. D1 goes on-hook.	**Failed**		**Connection Cleared**		
	• failedConnection	D1C1	• droppedConnection	D1C1	
	• failingDevice	D1	• releasingDevice	D1	
	• callingDevice	D1	• localConnectionInfo	connected	
	• calledDevice	D2	• cause	normalClr	
	• lastRedirectionDevice	NS	•		
	• localConnectionInfo	fail	•		
	• cause	blocked	•		
	• servicesPermitted	ClearConn(from HP4k V6 only!)	• servicesPermitted	ClearConn	
2. As a result of the "far end disconnect", the remaining connection D2C1 goes blocked.	**Connection Cleared**		**Failed**		
	• droppedConnection	D1C1	• failedConnection	D2C1	
	• releasingDevice	D1	• failingDevice	D2	
	• localConnectionInfo	null	• callingDevice	D1	
	• cause	normalClr	• calledDevice	D2	
	• servicesPermitted	none	• lastRedirectionDevice	NS	
			• localConnectionInfo	fail	
			• cause	blocked	
			• servicesPermitted	ClearConn	
3. The remaining device goes onhook.			**Connection Cleared**		
			• droppedConnection	D2C1	
			• releasingDevice	D2	
			• localConnectionInfo	null	
			• cause	normalClr	
			• servicesPermitted	none	

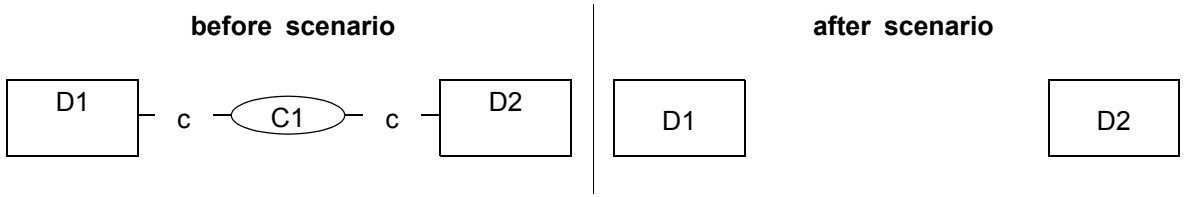
Table 5-10 SIP Device disconnects from a call by on-hook

Remark:

None

5.6.2 Device disconnects from a call using the Clear Connection service (remaining device goes blocked)

The Clear Connection service is used to disconnect device D1 from the call. After the service is invoked both devices go into blocked state.



See “Succesful answer call” on page 5-19 for the event flow to get into the “before service” state.

Activity	Monitored Device D1		Monitored Device D2		Comments
1. A Clear Connection service is invoked.	**ClearConnectionRequest** • connectionToBeCleared	D1C1			
2. Acknowledgement.	**ClearConnectionResult Response**				
3. D1C1 goes blocked.	**Failed** • failedConnection • failingDevice • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted	 D1C1 D1 D1 D2 NS fail blocked ClearConn	**Failed** • failedConnection • failingDevice • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted	 D1C1 D1 D1 D2 NS fail blocked ClearConn	This illustrates connection failures that report the Failed event for all devices involved with the call and that will provide a complete connectionID for the failed connection.
Scenario proceeds as shown in “Device disconnects from a call by on-hook” on page 5-20					

Table 5-11 Device disconnects by using the Clear Connection service

Remark:

The connection, on which the Connection Cleared service was initiated, first goes to failed state. This behaviour is different from the related scenario of ECMA TR/82.

5.7 External outgoing calls

Devices outside the CSTA sub-domain can not be directly monitored, network interface devices (NID) (e.g., trunk interface), act as proxies for those devices. Depending upon the type of signalling supported by the network, there may be a reduced level of event reporting after a Network Reached event and possibly no additional device feedback except connection clearing for trunks without Answer Supervision.

For external outgoing calls the associatedCalledDevice is a mandatory parameter. It specifies the Network Interface Device associated with the called device.

Remark:

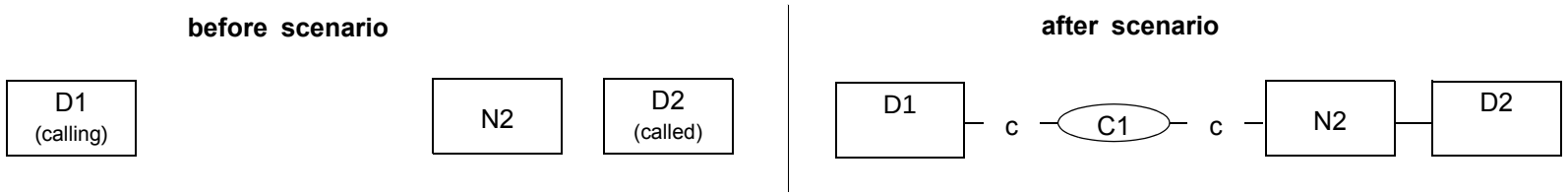
If the IP Direct Access (IPDA) feature is used with broken IP connection, as the Survivebility Path consists of normal network interface devices, the call between the Access Point and the central OpenScape switch will be reported as an external call.

5.7.1 Manual call to a device outside the CSTA subdomain

This scenario illustrates a manual external outgoing call.

Since device D2 is located outside the CSTA sub-domain, it can not be directly monitored through this CSTA interface and therefore no events will be seen for that device. However, device N2, which is a network interface device (NID) (e.g., trunk interface), acts as a proxy for device D2.

This scenario describes the behaviour of a network interface device with network information .



Activity	Monitored Device D1	Monitored Device N2	Comments
1. D1 goes offhook.	**Service Initiated** • initiatedConnection D1C1 • initiatingDevice D1 • localConnectionInfo initiated • cause normal • servicesPermitted ClearConn, DialDgt		

Table 5-12 External outgoing call (page 1 of 3)

Activity	Monitored Device D1	Monitored Device N2	Comments
2. **D1 completes dialling D2's number**	**Digits Dialled** • diallingConnection D1C1 • diallingDevice D1 • diallingSequence "1234" • localConnectionInfo initiated • cause normal • servicesPermitted none		D2's number is 1234
3. D1 is connected to the call.	**Originated** • originatedConnection D1C1 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn		
4. The call leaves the CSTA subdomain.	**Network Reached** • outbound connection N2C1 • NID device N2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted CallBack, ClearConn, SendUserInfo	**Network Reached** • outbound connection N2C1 • NID device N2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted Deflect, ClearConn, SendUserInfo	
5. Device D2 is alerted.	**Delivered** • connection N2C1 • alertingDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause networkSignal • assocCalled N2 • servicesPermitted CallBack, ClearConn, SendUserInfo	**Delivered** • connection N2C1 • alertingDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause networkSignal • assocCalled N2 • servicesPermitted Deflect, ClearConn, SendUserInfo	The cause of NetworkSignal indicates that the event is due to activity at the device located outside of the CSTA switching sub-domain (D2), not the NID (N2).

Table 5-12 External outgoing call (page 2 of 3)

Activity	Monitored Device D1		Monitored Device N2		Comments
6. Device D2 answers the call.	**Established**		**Established**		Network information is received from the network (this depends upon the type of signalling supported by the network).
	• establishedConnection	N2C1	• establishedConnection	N2C1	
	• answeringDevice	D2	• answeringDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	connected	
	• cause	networkSignal	• cause	networkSignal	
	• assocCalled	N2	• assocCalled	N2	
	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	• servicesPermitted	ClearConn, SendUserInfo	

Table 5-12 External outgoing call (page 3 of 3)

Remark:

The switching function provides the same event flow in the service initiated and in the manual case as well. It was modelled after the external outgoing Make Call service of ECMA TR/82.

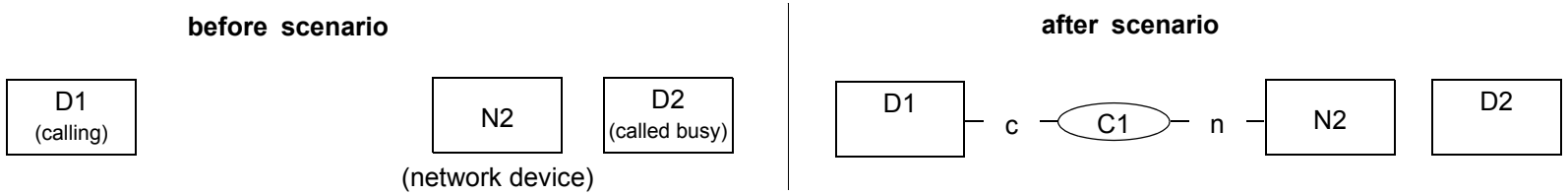
When Device D1 is not monitored and Device D2 has a call forward activated , then CA4000 will not be able to provide Device D2 as the originally called device. The originally called device will be the destination of the call forwarding.

5.7.2 Manual call to a busy device outside the CSTA subdomain

This scenario illustrates a manual external outgoing call to a busy device.

Since device D2 is located outside the CSTA sub-domain, it can not be directly monitored through this CSTA interface and therefore no events will be seen for that device. However, device N2, which is a network interface device (NID) (e.g., trunk interface), acts as a proxy for device D2.

This scenario describes the behaviour of a network interface device with network information .



Activity	Monitored Device D1		Monitored Device N2		Comments
Steps 1-3 are shown in "Manual call to a device outside the CSTA subdomain" on page 5-24.					
4. D2 is busy. The call can not be completed. D1 hears busy tone.	**Failed**		**Failed**		
	• failedConnection	N2C1	• failedConnection	N2C1	
	• failingDevice	D2	• failingDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	fail	
	• cause	busy	• cause	busy	
	• assocCalled	N2	• assocCalled	N2	
	• servicesPermitted	ClearConn	• servicesPermitted	none	
5. The busy connection is cleared immediately.	**Connection Cleared**		**Connection Cleared**		
	• droppedConnection	N2C1	• droppedConnection	N2C1	
	• releasingDevice	N2	• releasingDevice	N2	
	• localConnectionInfo	connected	• localConnectionInfo	null	
	• cause	normalClr	• cause	normalClr	
	• servicesPermitted	ClearConn	• servicesPermitted	none	

Table 5-13 External outgoing call to a busy device

Remark:

The protocol converter of the switching function will immediately send a Connection Cleared event after a connection goes into the busy failed state. This does not necessarily mean, that the connection physically goes to idle.

Possible event causes:

Event Cause	Description	Associated Features
Busy	The call failed after it encountered a busy or unavailable device.	Connection Failure
Destination Out of Order	The call failed because it encountered a destination out of service.	Connection Failure
Do Not Disturb	The call failed because it encountered a device that has the do not disturb feature set.	Do Not Disturb, Call Forwarding
Invalid Number Format	The call failed because the dialled number is incorrect.	Connection Failure
Network Congestion	The call failed because it encountered a congested network. In some circumstances, this event cause indicates that the user is listening to a special signal tone from a network. The tone may be accompanied by a voiced statement similar to “All circuits are busy...”	Connection Failure
Network Signal	The call failed because it encountered a problem after it left the switching sub-domain.	External Calls
Number Unallocated	The call failed because the called number is not allocated to a subscriber.	Connection Failure

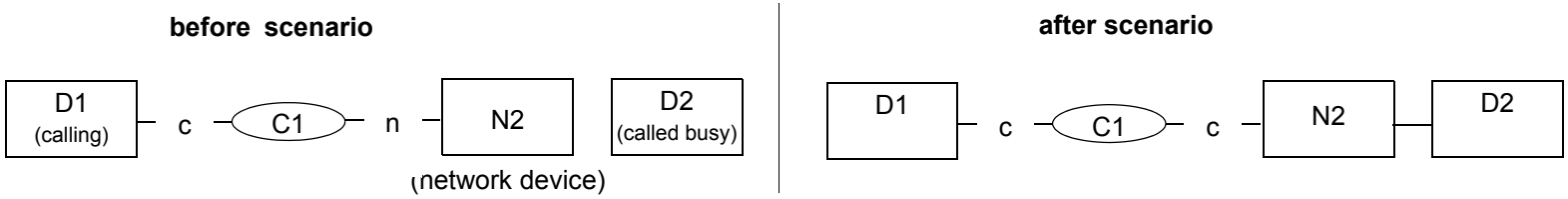
Table 5-14 Event causes

5.7.3 External outgoing camp-on

This scenario illustrates an automatic camp-on to a busy device.

Since device D2 is located outside the CSTA sub-domain, it can not be directly monitored through this CSTA interface and therefore no events will be seen for that device. However, device N2, which is a network interface device (NID) (e.g., trunk interface), acts as a proxy for device D2.

This scenario describes the behaviour of a network interface device with network information .



See “Manual call to a busy device outside the CSTA subdomain” on page 5-26 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1		Monitored Device N2		Comments
1. The call leaves the CSTA subdomain.	**Network Reached**		**Network Reached**		
	• outbound connection	N2C1	• outbound connection	N2C1	
	• NID device	N2	• NID device	N2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	connected	
	• cause	normal	• cause	normal	
	• servicesPermitted	CallBack, ClearConn, SendUserInfo	• servicesPermitted	Deflect, ClearConn, SendUserInfo	
2. Device D1 hears ringback, and the call queues to D2 at the other end.	**Delivered**		**Delivered**		The switching function provides Delivered event. It means that an application at the outgoing side will not be able to differentiate the camp on call from a basic call.
	• connection	N2C1	• connection	N2C1	
	• alertingDevice	D2	• alertingDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	connected	
	• cause	networkSignal	• cause	networkSignal	
	• assocCalled	N2	• assocCalled	N2	
	• servicesPermitted	CallBack, ClearConn, SendUserInfo	• servicesPermitted	Deflect, ClearConn, SendUserInfo	

Table 5-15 External outgoing camp-on

Remark:

None

5.8 External incoming calls

Devices outside the CSTA sub-domain can not be directly monitored, network interface devices (NID) (e.g., trunk interface), act as proxies for those devices. Depending upon the type of signalling supported by the network, the following information can be present :

- networkCallingDevice: It specifies the ANI number if it is provided by the network. Otherwise it is not present.
- callingDevice: It specifies the ANI number if it is provided by the network. Otherwise it is not known (NK).
- networkCalledDevice: It specifies the DNIS number if it is provided by the network. Otherwise it is not present. This information element will be only present in case of a DNIS trunk.
- calledDevice: It specifies an internal format of the DNIS number if it is provided by the network. Otherwise it is not known (NK) .
- assocCalledDevice: It specifies an internal format of the DNIS number, that is different from the calledDevice, if it is provided by the network. Otherwise it is not present.

For external incoming calls the associatedCallingDevice is a mandatory parameter. It specifies the Network Interface Device associated with the calling device if the call is incoming.

Remark:

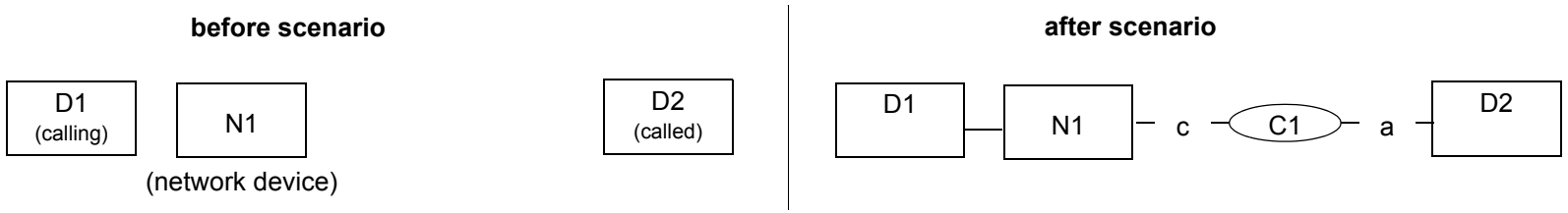
If the IP Direct Access (IPDA) feature is used with broken IP connection, as the Survivebility Path consists of normal network interface devices, the call between the Access Point and the central OpenScape switch will be reported as an external call.

5.8.1 External incoming call

This scenario illustrates a manual external incoming call.

Since device D1 is located outside this CSTA sub-domain, it can not be directly monitored through this CSTA interface and therefore no events will be seen for that device. However, device N1, which is a network interface device (NID) (e.g., trunk interface), acts as a proxy for device D1.

This scenario describes the behaviour of a typical non-DNIS network interface device with ANI network information.



Activity	Monitored Device N1		Monitored Device D2	Comments
1. Indicates an incoming call from N1.	**Service Initiated**			
	• initiatedConnection	N1C1		
	• initiatingDevice	N1		
	• localConnectionInfo	initiated		
	• cause	normal		
	• assocCalling	N1		
	• servicesPermitted	ClearConn		
2. The NID has connected to the call.	**Originated**			
	• originatedConnection	N1C1		
	• callingDevice	D1		
	• calledDevice	D2		
	• lastRedirectionDevice	NS		
	• localConnectionInfo	connected		
	• cause	normal		
	• assocCalling	N1		
	• networkCalling	D1		
	• servicesPermitted	ClearConn		

Table 5-16 External incoming call (page 1 of 2)

Activity	Monitored Device N1		Monitored Device D2		Comments
3. Device D2 is available and starts ringing.	**Delivered**		**Delivered**		
	• connection	D2C1	• connection	D2C1	
	• alertingDevice	D2	• alertingDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• origNID connID	N1C1	• origNID connID	N1C1	
	• localConnectionInfo	connected	• localConnectionInfo	alert	
	• cause	normal	• cause	normal	
	• assocCalling	N1	• assocCalling	N1	
	• networkCalling	D1	• networkCalling	D1	
	• servicesPermitted	ClearConn, SendUserInfo	• servicesPermitted	Answer, ClearConn, Deflect, SendUserInfo	

Table 5-16 External incoming call (page 2 of 2)

Remark:

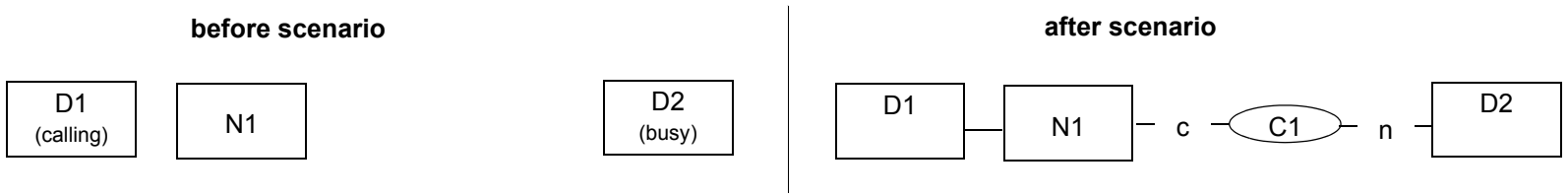
None

5.8.2 Incoming external call to a busy device

This scenario illustrates a manual external incoming call to a busy device.

Since device D1 is located outside this CSTA sub-domain, it can not be directly monitored through this CSTA interface and therefore no events will be seen for that device. However, device N1, which is a network interface device (NID) (e.g., trunk interface), acts as a proxy for device D1.

This scenario describes the behaviour of a typical non-DNIS network interface device with ANI network information.



Activity	Monitored Device N1		Monitored Device D2		Comments
Steps 1-2 are shown in "External incoming call" on page 5-30.					
3. D2 is busy. The call can not be completed. D1 hears busy tone.	**Failed** • failedConnection • failingDevice • callingDevice • calledDevice • lastRedirectionDevice • origNID connID • localConnectionInfo • cause • assocCalling • networkCalling • servicesPermitted	 D2C1 D2 D1 D2 NS N1C1 connected busy N1 D1 ClearConn	**Failed** • failedConnection • failingDevice • callingDevice • calledDevice • lastRedirectionDevice • origNID connID • localConnectionInfo • cause • assocCalling • networkCalling • servicesPermitted	 D2C1 D2 D1 D2 NS N1C1 fail busy N1 D1 none	
4. The busy connection is cleared immediately.	**Connection Cleared** • droppedConnection • releasingDevice • localConnectionInfo • cause • servicesPermitted	 D2C1 D2 connected normalClr ClearConn	**Connection Cleared** • droppedConnection • releasingDevice • localConnectionInfo • cause • servicesPermitted	 D2C1 D2 null normalClr none	

Table 5-17 External incoming call

Remark:

The protocol converter of the switching function will immediately send a Connection Cleared event after a connection goes into the busy failed state. This does not necessarily mean, that the connection physically goes to idle.

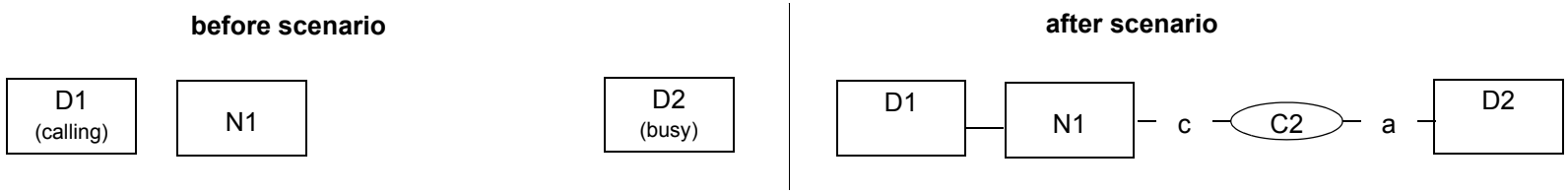
The Network Interface Device behaviour differs from the normal extensions in the clearing of the busy connection, since the Network Interface Device will not be blocked after this situation. It means that no Failed event will be provided.

5.8.3 External incoming camp-on

This scenario illustrates an incoming call camp on to a busy device. The calling device D1 is located outside the CSTA sub-domain. This feature queues the call for the busy device D2 until that device becomes available.

Since device D1 is located outside this CSTA sub-domain, it can not be directly monitored through this CSTA interface and therefore no events will be seen for that device. However, device N1, which is a network interface device (NID) (e.g., trunk interface), acts as a proxy for device D1.

This scenario describes the behaviour of a typical non-DNIS network interface device with ANI network information.



Activity	Monitored Device N1		Monitored Device D2	Comments
1. Indicates an incoming call from N1.	**Service Initiated**			
	• initiatedConnection	N1C2		
	• initiatingDevice	N1		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	ClearConn		
2. The NID is connected to the call.	**Originated**			
	• originatedConnection	N1C2		
	• callingDevice	D1		
	• calledDevice	D2		
	• lastRedirectionDevice	NS		
	• localConnectionInfo	connected		
	• cause	normal		
	• assocCalling	N1		
	• networkCalling	D1		
	• servicesPermitted	ClearConn		

Table 5-18 External incoming call (page 1 of 2)

Activity	Monitored Device N1		Monitored Device D2		Comments
3. D2 is busy. The call is queued at D2.	**Queued**		**Queued**		At the incoming side the computing function gets the necessary information to identify the camp on, not like at the outgoing side.
	• queuedConnection	D2C2	• queuedConnection	D2C2	
	• queue	D2	• queue	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	queue	
	• cause	campOn	• cause	campOn	
	• assocCalling	N1	• assocCalling	N1	
	• networkCalling	D1	• networkCalling	D1	
	• servicesPermitted	ClearConn, SendUserInfo	• servicesPermitted	SendUserInfo	
4. Device D2 sometime later clears from its active call.			**Connection Cleared**		C1 was the previous active call of D2 and the reason why it was busy.
			• droppedConnection	D2C1	
			• releasingDevice	D2	
			• localConnectionInfo	null	
			• cause	normalClr	
			• servicesPermitted	none	
5. Since D2 is available, the call alerts D2.	**Delivered**		**Delivered**		
	• connection	D2C2	• connection	D2C2	
	• alertingDevice	D2	• alertingDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• origNID connID	N1C2	• origNID connID	N1C2	
	• localConnectionInfo	connected	• localConnectionInfo	alert	
	• cause	recall	• cause	recall	
	• assocCalling	N1	• assocCalling	N1	
	• networkCalling	D1	• networkCalling	D1	
	• servicesPermitted	ClearConn, SendUserInfo	• servicesPermitted	Answer, ClearConn, Deflect, SendUserInfo	

Table 5-18 External incoming call (page 2 of 2)

Remark:

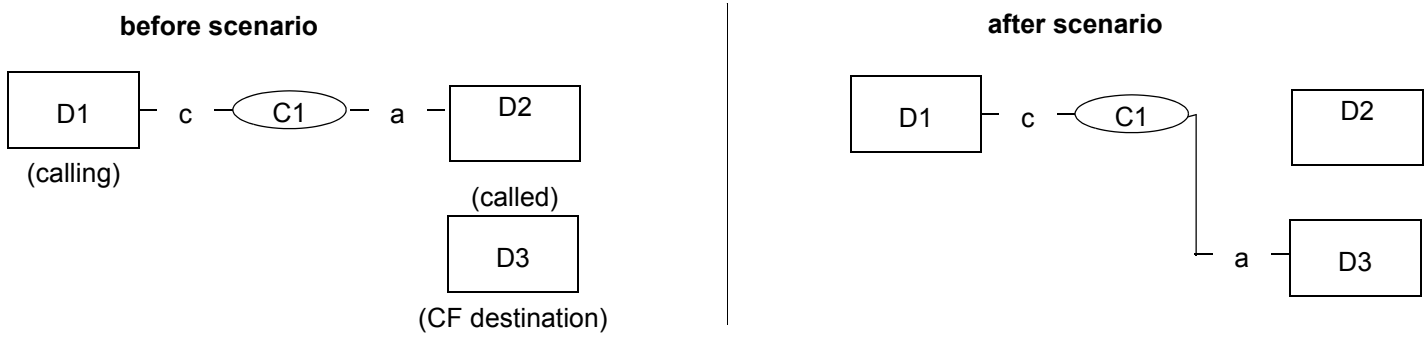
None

5.9 Forwarding Call Scenarios

The bsasic modell of the switching function is to provide the Diverted event only to the device which leaves the call, there is an optional possibility to configure it to provide Diverted event also for the calling side in all scenarios where diversion involved. The local connection info remains the calling device's actual local connection info (connected). Services permitted are not provided in Diverted.

5.9.1 Call Forward - No Answer

This scenario illustrates the event flow of a basic call forward no answer. A call comes to a device which is set to forward calls to a predefined device after a specified number of rings / time.



See "Manually dialled call" on page 5-4 for the event flow to get into the "before scenario" state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. D2 is alerted for a specified number of rings and then forwards the call to device D3.		Diverted • connection D2C1 • divertingDevice D2 • newDestination D3 • Calling D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo null • cause forwardNoAnswer • servicesPermitted none		Device D3 is the device predefined by device D2 to forward its call. The switching function sends the Diverted event only to the divertingDevice.
2. D3 starts ringing.	Delivered • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D2 • localConnectionInfo connected • cause forwardNoAnswer • servicesPermitted CallBack, ClearConn, SendUserInfo		Delivered • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D2 • localConnectionInfo alert • cause forwardNoAnswer • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo	

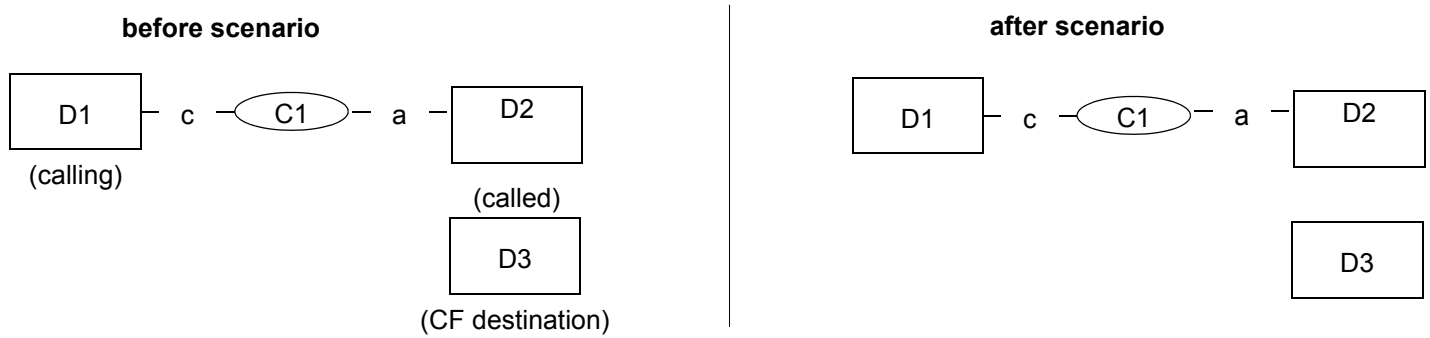
Table 5-19 Call Forward - No Answer

Remark:

None

5.9.2 Call Forward - No Answer: Forwarding device and Destination have Offered Mode on; Destination is busy

This scenario illustrates the event flow of a basic call forward no answer. A call comes to a device which is set to forward calls to a predefined device after a specified number of rings / time. Normally this monitoring type requires Divered and Offered events also for calling side. It is configurable.



See “Manually dialled call” on page 5-4 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. D2 is alerted for a specified number of rings and then forwards the call to device D3.	Diverted (optional) • connection D2C1 • divertingDevice D2 • newDestination D3 • Calling D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause forwardNoAnswer • servicesPermitted none	Diverted • connection D2C1 • divertingDevice D2 • newDestination D3 • Calling D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo null • cause forwardNoAnswer • servicesPermitted none		Device D3 is the device predefined by device D2 to forward its call. The switching function sends the Diverted event only to the divertingDevice.
2. Call is offered to D3	Offered (optional) • offeredConnection D3C1 • offeredDevice D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D2 • localConnectionInfo connected • cause forwardNoAnswer • servicesPermitted ClearConn		Offered • offeredConnection D3C1 • offeredDevice D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D2 • localConnectionInfo alerting • cause forwardNoAnswer • servicesPermitted AcceptCall, ClearConn, Deflect	
3. Application accepts the call			Accept Cal lRequest • callToBeAccepted D3C1	
4. Acknowledged			Accept Call Response	

Table 5-20 Call Forward - No Answer to Destination with Offered Mode(page 1 of 2)

Activity	Monitored Device D1		Monitored Device D2		Monitored Device D3		Comments
5. Call fails on D3	Diverted (optional) • divertedConnection • divertingDevice • newDest • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted	 D3C1 D3 D2 D1 D2 NS connected redirected none			Diverted • divertedConnection • divertingDevice • newDest • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted	 D3C1 D3 D2 D1 D2 NS null redirected none	
6. D2 starts to alert again	Delivered • deliveredConnection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted	 D2C1 D2 D1 D2 D3 connected redirected callBack, clearConn	Delivered • deliveredConnection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted	 D2C1 D2 D1 D2 NS alerting redirected answer, clearConn, deflect			

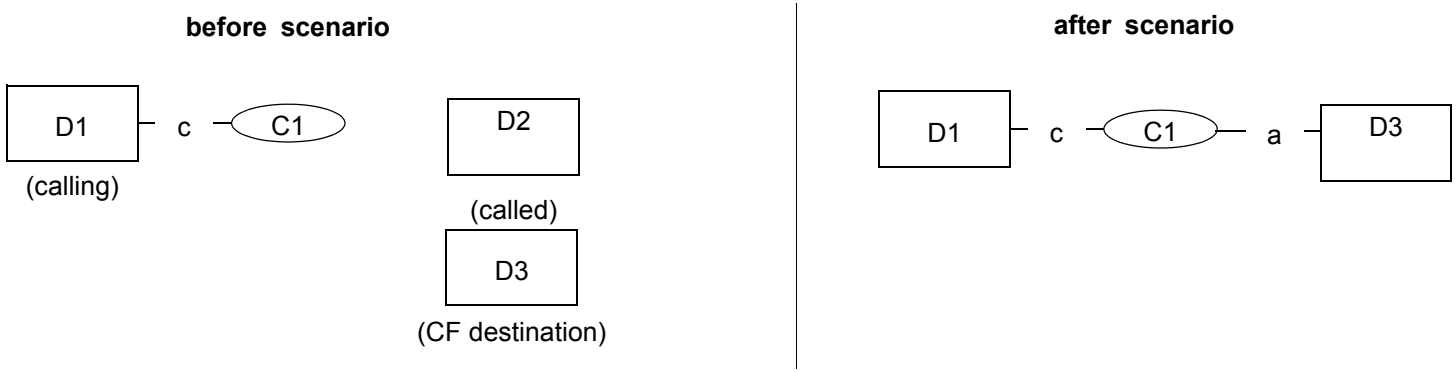
Table 5-20 Call Forward - No Answer to Destination with Offered Mode(page 2 of 2)

Remark:

None

5.9.3 Call Forward - Immediate

This scenario illustrates the flow for a basic call forward immediate. A call comes to a device which is set to forward calls immediately to a predefined device.



See “Manually dialled call” on page 5-4 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
		Diverted • divertingConnection D2C1 • divertingDevice D2 • newDestination D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDev NS • localConnectionInfo null • cause forwardImm • servicesPermitted none		
1. D3 starts ringing.	Delivered • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D2 • localConnectionInfo connected • cause forwardImmediate • servicesPermitted CallBack, ClearConn, SendUserInfo		Delivered • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D2 • localConnectionInfo alert • cause forwardImmediate • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo	

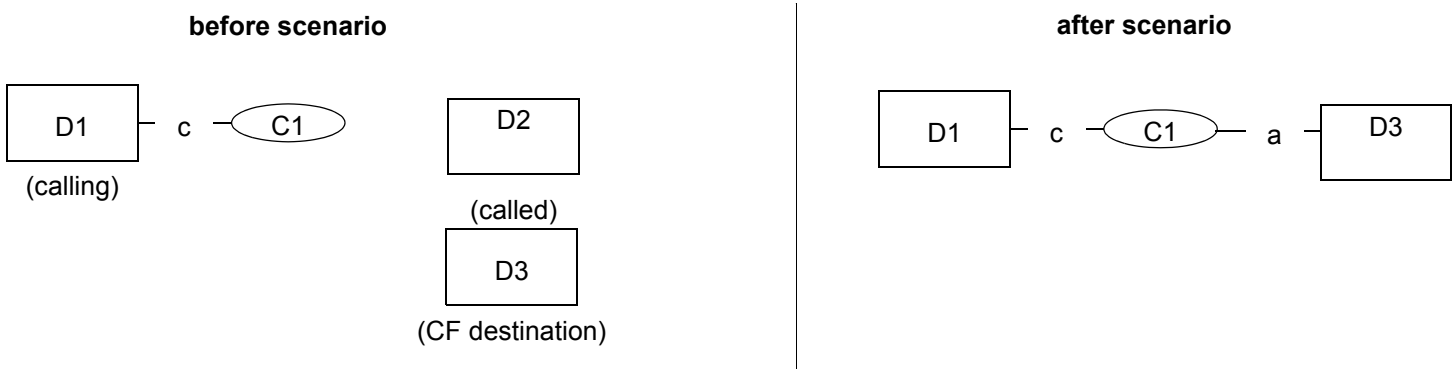
Table 5-21 Call Forward - Immediate

Remark:

None

5.9.4 Call Forward - Immediate - Called Party is OOS (Out Of Service)

This scenario illustrates the flow for a basic call forward immediate. A call comes to a device which is out of service and set to forward calls immediately to a predefined device.



See “Manually dialled call” on page 5-4 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2 is OOS	Monitored Device D3	Comments
2. Called party D2 is Out of Service and D2 has Cf-All to D3		Diverted • divertingConnection D2C1 • divertingDevice D2 • newDestination D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDev NS • localConnectionInfo null • cause forwardImm • servicesPermitted none		

Table 5-22 Call Forward Immediate Called Party OutOfService (OOS)

Activity	Monitored Device D1	Monitored Device D2 is OOS	Monitored Device D3	Comments
3. D3 starts ringing.	Delivered • connection D3C1 • alertingDevice D3 • callingDevice D1 • caledDevice D2 • lastRedirectionDev D2 • localConnectionInfo connected • cause forwardImm • servicesPermitted CallBack, Clear Conn, SendUI		Delivered • connection D3C1 • alertingDevice D3 • callingDevice D1 • caledDevice D2 • lastRedirectionDev D2 • localConnectionInfo alert • cause forwardImm • servicesPermitted Answer, Clear Conn, Deflect, SendUI	

Table 5-22 Call Forward Immediate Called Party OutOfService (OOS)

Remark:

None

5.9.5 Call Forward - Busy

In case of a Call forward busy similar event flow will be generated to the Call forward immediate case.

See “Call Forward - Immediate” on page 5-40.

The only difference is the CSTA event cause which will be forwardbusy in this case.

Remark:

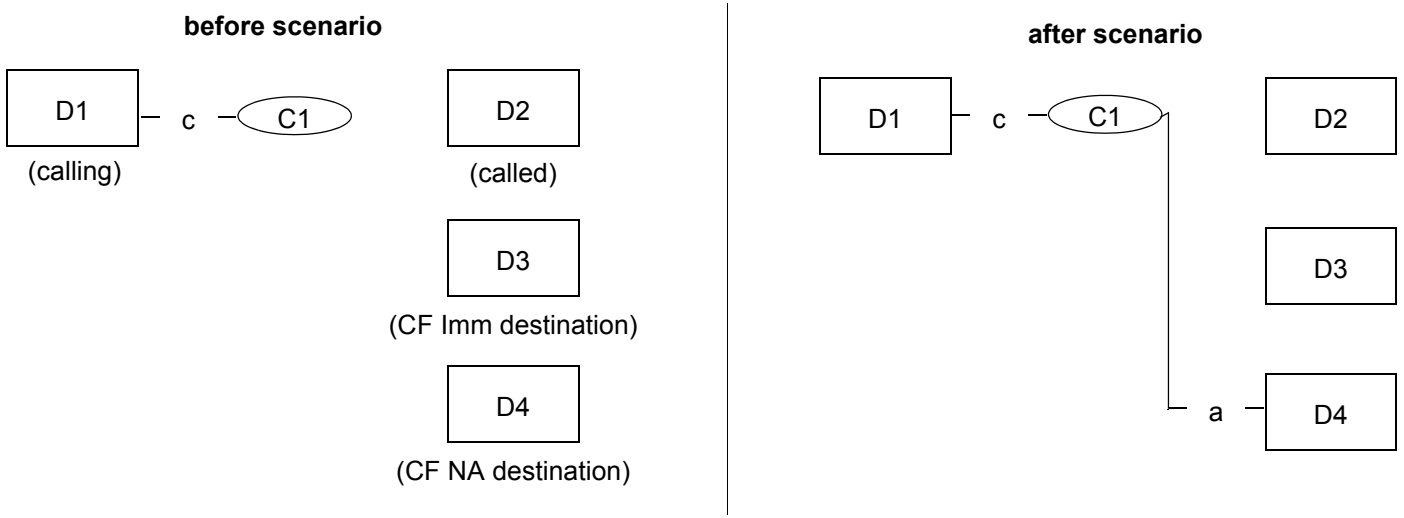
None

5.10 Multiple Forwarding Scenarios

5.10.1 Call Forward Immediate followed by Call Forward No Answer

This scenario illustrates a multiple forwarding call scenario.

A call comes to a device which is set to forward every incoming call to a predefined device immediately. The destination device is set to forward calls to another predefined device after a specified number of rings.



See “Manually dialled call”, on page 5-4 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Monitored Device D4	Comments
1. Device D2 has CF All to D3.		Diverted • connection D2C1 • divertingDevice D2 • newDestination D3 • Calling D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo null • cause forwardImm • servicesPermitted none			

Table 5-23 Call Forward Immediate followed by Call Forward No Answer

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Monitored Device D4	Comments
2. D3 starts ringing.	Delivered • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NK • localConnectionInfo connected • cause forwardImmediate • servicesPermitted CallBack, ClearConn, SendUserInfo		Delivered • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NK • localConnectionInfo alert • cause forwardImmediate • servicesPermitted Answer,ClearConn, Deflect, SendUserInfo		The switching function supports the “Forwarding is Triggered before the Call is Delivered the Device” CSTA modelling option. That is why lastRedirectionDevice is always NK in this case
3. D3 is alerted for a specified number of rings then forwards the call to device D4.			Diverted • connection D3C1 • divertingDevice D3 • newDestination D4 • Calling D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo null • cause forwardNoAnswer • servicesPermitted none		Device D4 is the device predefined by device D3 to forward its call. The switching function sends the Diverted event only to the diverting Device

Table 5-23 Call Forward Immediate followed by Call Forward No Answer

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Monitored Device D4	Comments
4. D4 starts ringing.	Delivered • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D3 • localConnectionInfo connected • cause forwardNoAnswer • servicesPermitted CallBack, ClearConn, SendUserInfo			Delivered • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D3 • localConnectionInfo alert • cause forwardNoAnswer • servicesPermitted Answer,ClearConn, Deflect, SendUserInfo	

Table 5-23 Call Forward Immediate followed by Call Forward No Answer

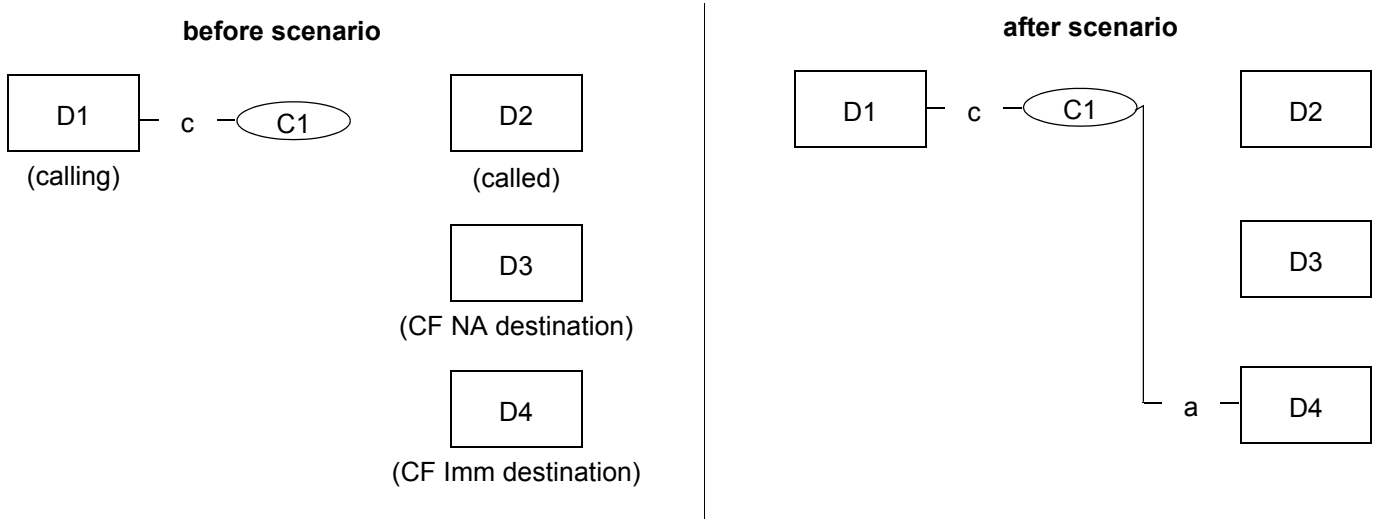
Remark: none

5.10.2 Call Forward No Answer followed by Call Forward Immediate

5.10.2.1 Destination is available

This scenario illustrates a multiple forwarding call scenario.

A call comes to a device which is set to forward every incoming call to a predefined device after a specified number of rings. The destination device is set to forward calls to another predefined device immediately. D4 is available.



See “Manually dialled call”, on page 5-4 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Devicr D3	Monitored Device D4	Comments
1. D2 starts ringing.	Delivered • connection D2C1 • alertingDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted CallBack, ClearConn, SendUserInfo	Delivered • connection D2C1 • alertingDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo alert • cause normal • servicesPermitted Answer,ClearConn, Deflect, SendUserInfo			

Table 5-24 Call Forward No Answer followed by Call Forward Immediate (page 1 of 3)

Activity	Monitored Device D1	Monitored Device D2	Monitored Devicr D3	Monitored Device D4	Comments
2. D2 is alerted for a specified number of rings then tryes to reach device D3. D3 has a call forwarding activated to D4.			Diverted • connection D3C1 • divertingDevice D3 • newDestination D4 • Calling D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo null • cause forwardImm • servicesPermitted none		The switching function sends the Diverted event only to the divertingDevice.
3. If D4 is available then the call will be forwarded from D2 to D4.		Diverted • connection D2C1 • divertingDevice D2 • newDestination D4 • Calling D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo null • cause forwardNoAnswer • servicesPermitted none			The switching function sends the Diverted event only to the divertingDevice.

Table 5-24 Call Forward No Answer followed by Call Forward Immediate (page 2 of 3)

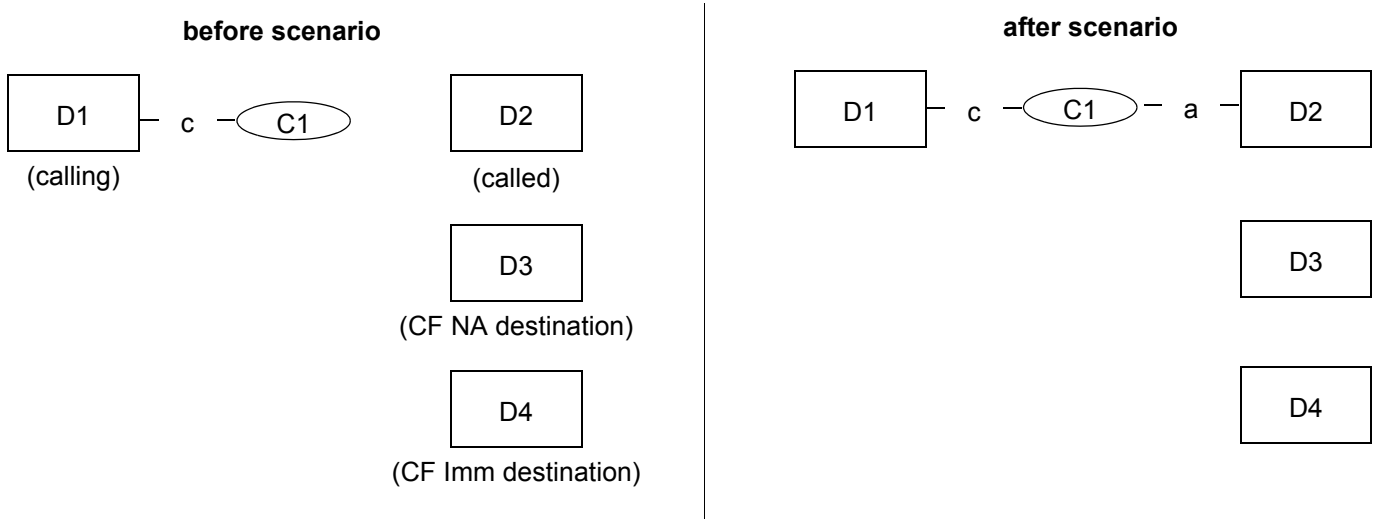
Activity	Monitored Device D1	Monitored Device D2	Monitored Devicr D3	Monitored Device D4	Comments
4. D4 starts ringing.	Delivered • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D4 • localConnectionInfo connected • cause forwardNoAnswer • servicesPermitted CallBack, ClearConn, SendUserInfo			Delivered • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D4 • localConnectionInfo alert • cause forwardNoAnswer • servicesPermitted Answer,ClearConn, Deflect, SendUserInfo	Because the Call Forward No Answer happens after the Call Forward Immediate, the lastRedirectionDevice is D2.

Table 5-24 Call Forward No Answer followed by Call Forward Immediate (page 3 of 3)

Remark: Please note that the Call Forward No Answer happens physically after the Call Forward Immediate!

5.10.2.2 Destination is not available

A call comes to a device which is set to forward every incoming call to a predefined device after a specified number of rings. The destination device is set to forward calls to another predefined device immediately. D4 is not available



See “Manually dialled call”, on page 5-4 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Devicr D3	Monitored Device D4	Comments
1. D2 starts ringing.	Delivered • connection D2C1 • alertingDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted CallBack, ClearConn, SendUserInfo	Delivered • connection D2C1 • alertingDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo alert • cause normal • servicesPermitted Answer,ClearConn, Deflect, SendUserInfo			

Table 5-25 Call Forward No Answer followed by Call Forward Immediate (page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Devicr D3	Monitored Device D4	Comments
2. D3 forwards the call to D4.			Diverted • connection D3C1 • divertingDevice D3 • newDestination D4 • Calling D1 • calledDevice D2 • lastRedirection Device NS • localConnectionInfo null • cause forwardImm • servicesPermitted none		D3 has Call Forward Immediate activated to device D4. That is why newDestination is D4. The switching function sends the Diverted event only to the divertingDevice.
3. D4 is busy in another call.				Failed • connection D4C1 • failingDevice D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo null • cause destinationNotAvailable • servicesPermitted none	Because of the Call Forward Immediate from D3 to D4, the lastRedirectionDevice is NK.

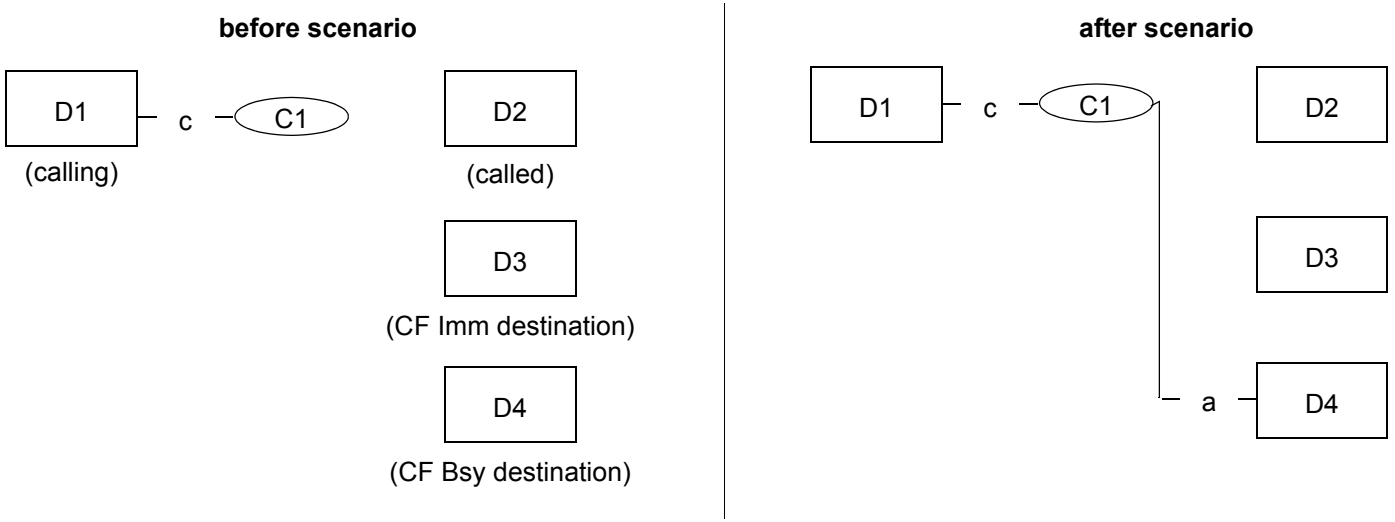
Table 5-25 Call Forward No Answer followed by Call Forward Immediate (page 2 of 2)

Remark: In this case Device D2 will continue ringing, so the Call Forward No Answer will not be executed. After the next timeout the switch tries to reach D3 and D4 again (periodically) until D4 becomes available or D2 answers the call.

5.10.3 Call Forward Immediate followed by Call Forward Busy

This scenario illustrates a multiple forwarding call scenario.

A call comes to a device which is set to forward every incoming call to a predefined device immediately. The destination device is set to forward calls to another predefined device in case of it is busy in another call.



See “Manually dialled call”, on page 5-4 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Monitored Device D4	Comments
1. D2 forwards the call to D3.		Diverted • connection D2C1 • divertingDevice D2 • newDestination D3 • Calling D1 • calledDevice D2 • lastRedirection Device NS • localConnectionInfo null • cause forwardImm • servicesPermitted none			

Table 5-26 Call Forward Immediate followed by Call Forward Busy

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Monitored Device D4	Comments
2. D3 forwards the call to D4.			Diverted • connection D3C1 • divertingDevice D3 • newDestination D4 • Calling D1 • calledDevice D2 • lastRedirectionDevice D2 • localConnectionInfo null • cause forwardBusy • servicesPermitted none		D3 is busy in another call.
3. D4 starts ringing.	Delivered • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NK • localConnectionInfo connected • cause forwardBusy • servicesPermitted CallBack, ClearConn, SendUserInfo			Delivered • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NK • localConnectionInfo alert • cause forwardBusy • servicesPermitted Answer,ClearConn, Deflect, SendUserInfo	

Table 5-26 Call Forward Immediate followed by Call Forward Busy

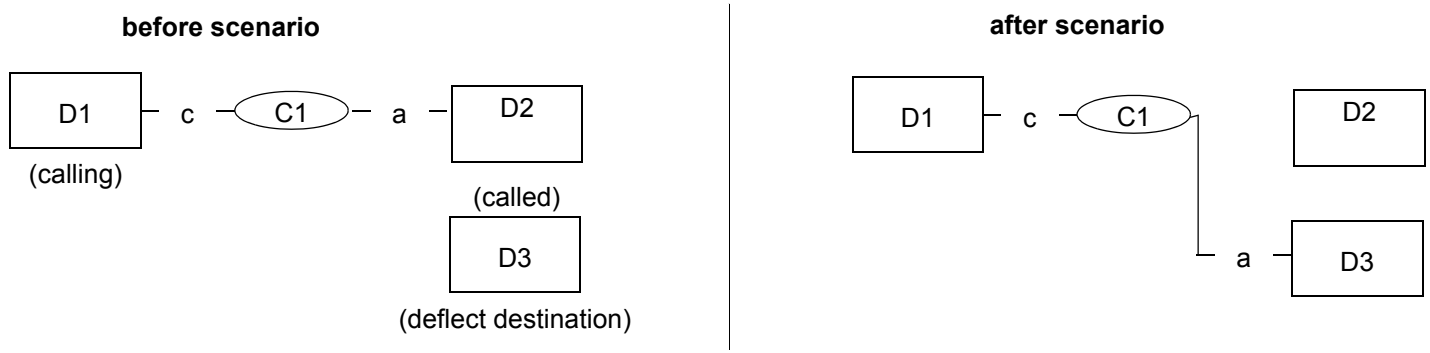
Remark: none

5.11 Call Movement Scenarios

This section includes examples of moving calls from one device to another, initiated manually or by CSTA services.

5.11.1 Deflect Call service

This scenario illustrates how an alerting call is diverted to another destination.



See “Manually dialled call” on page 5-4 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Deflect service is invoked on behalf of D2.		Deflect Call Request • connection to be deflected: D2C1 • new destination D3		
2. Acknowledgement.		Deflect Call Response		
3. The event indicates that the call has been diverted from D1.		Diverted • connection D2C1 • divertingDevice D2 • newDestination D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo null • cause redirected • servicesPermitted none		The switching function sends the Diverted event only to the divertingDevice.

Table 5-27 Deflect Call service(page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
4. D3 starts ringing.	Delivered • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D2 • localConnectionInfo connected • cause redirected • servicesPermitted CallBack, ClearConn, SendUserInfo		Delivered • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D2 • localConnectionInfo alert • cause redirected • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo	

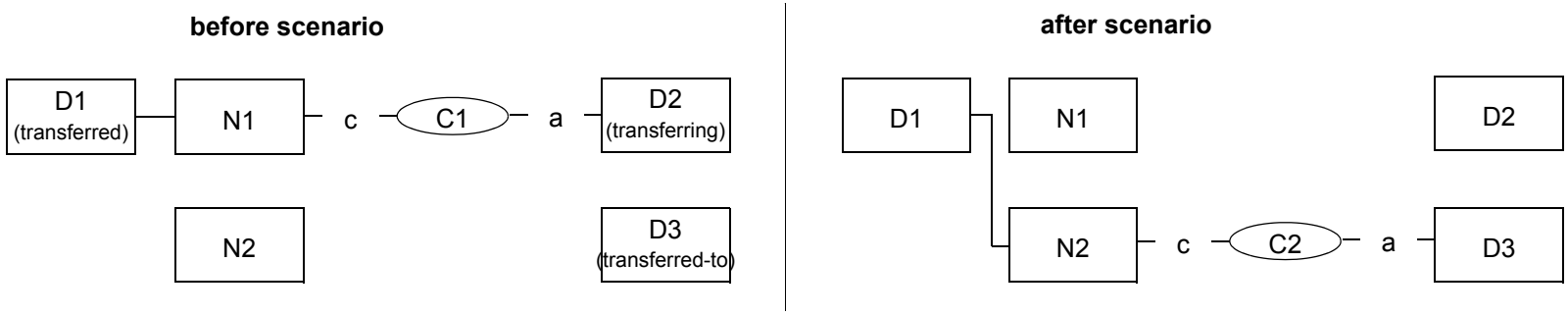
Table 5-27 Deflect Call service(page 2 of 2)

Remark:

None

5.11.2 Deflect call with ReRouting enabled

This scenario illustrates a successful Deflect Service when ReRouting is enabled.



Activity	Monitored Device N1	Monitored Device D2	Monitored Device N2	Monitored Device D3	Comments
1. Deflect Call service is invoked on behalf of device D1.	Deflect Call Request • connToBeDeflect D2C1 • newDestinationDevice D3				
2. Acknowledgement.	Deflect Call Response				
3. New network device is seized incoming.			Service Initiated • initiatedConnection N2C2 • initiatingDevice N1 • localConnectionInfo initiated • cause normal • servicesPermitted ClearConn		
4. N2 is connected to the call.			Originated • originatedConnection N2C2 • callingDevice D1 • calledDevice D3 • localConnectionInfo connected • cause normal • servicesPermitted ClearConn • networkCallingDevice D1 • assocCallingDevice N2		

Table 5-28 Deflect call with ReRouting enabled (page 1 of 2)

Activity	Monitored Device N1	Monitored Device D2	Monitored Device N2	Monitored Device D3	Comments
5. The call alerts at D3.			Delivered • connection D3C2 • alertingDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NS • origNID N2C2 • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, SendUserInfo • networkCallingDevice D1 • assocCallingDevice N2	Delivered • connection D3C2 • alertingDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NS • origNID N2C2 • localConnectionInfo alerting • cause normal • servicesPermitted Answer, ClearConn, Deflect, SendUser-Info • networkCallingDevice D1 • assocCallingDevice N2	
6. The network device N1 clears from the call.	Connection Cleared • droppedConnection N1C1 • releasingDevice N1 • localConnectionInfo null • cause normalClr • servicesPermitted none	Connection Cleared • droppedConnection N1C1 • releasingDevice N1 • localConnectionInfo alerting • cause normalClr • servicesPermitted none			
7. Device D2 clears also.		Connection Cleared • droppedConnection D2C1 • releasingDevice D2 • localConnectionInfo null • cause normalClr • servicesPermitted none			

Table 5-28 Deflect call with ReRouting enabled (page 2 of 2)

Remark:

When ReRouting is disabled, the call flow will be similar to “Deflect Call service” on page 5-53.

In order to disable RE-ROUTING you need to remove the FNAN (RWSN in german) parameter from the COT AMO of the incoming trunk. This parameter enables the re-routing for Call Forward No Answer and Deflect scenarios.

English AMO version:

CHA-COT:COTNO=<number>,COTTYPE=COTDEL,PAR=FNAN;

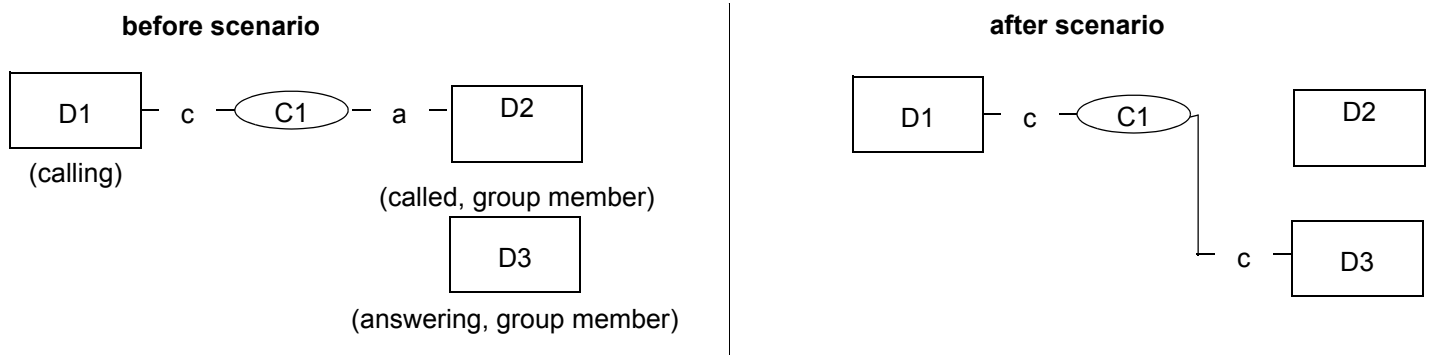
German AMO version:

AE-COT:COTNU=<number>,COTART=COTWEG,PAR=RWSN;

5.11.3 Manual group pickup

This scenario illustrates a pickup of a call that is alerting at a device as a member of a pickup group.

The call is moved and connected at the new specified destination.



See “Manually dialled call” on page 5-4 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. D3 hits the pickup key.		Diverted • connection D2C1 • divertingDevice D2 • newDestination D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo null • cause callPickup • servicesPermitted none		The switching function sends the Diverted event only to the divertingDevice.

Table 5-29 Manual group pickup(page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
2. D3 is in connection with D1.	Established • establishedConnection D3C1 • answeringDevice D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause CallPickup • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo		Established • establishedConnection D3C1 • answeringDevice D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause CallPickup • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	The switching function provides lastRedirectionDevice NS instead of the proper value.

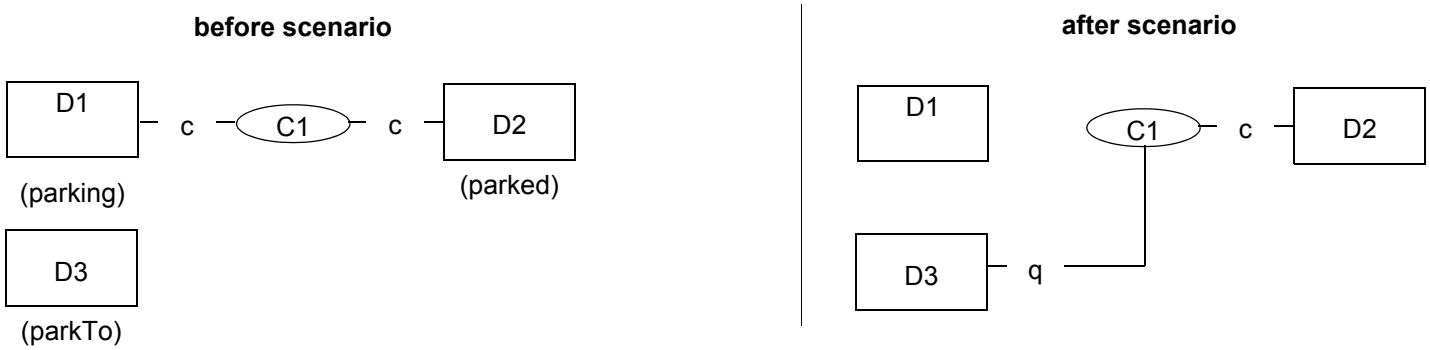
Table 5-29 Manual group pickup(page 2 of 2)

Remark:

None

5.11.4 Manual directed park call

This scenario illustrates how a connected call is parked at another device.



See “Succesful answer call” on page 5-19 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. D1 presses Park key. Connection is placed on hold.	Held • heldConnection D1C1 • holdingDevice D1 • localConnectionInfo held • cause consultation • servicesPermitted SendUserInfo, Reconnect	Held • heldConnection D1C1 • holdingDevice D1 • localConnectionInfo connected • cause consultation • servicesPermitted ClearConn, SendUserInfo		
2. D1 dials park destination.	Service Initiated • initiatedConnection D1C2 • initiatingDevice D1 • localConnectionInfo initiated • cause consultation • servicesPermitted ClearConn, DialDgt, Reconnect			
3. **D1 completes dialling destination's number ("1234")**	Digits Dialled • diallingConnection D1C1 • diallingDevice D1 • diallingSequence "1234" • localConnectionInfo initiated • cause normal			
4. The call has been diverted from D1.	Diverted • connection D1C1 • divertingDevice D1 • newDestination D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo null • cause park • servicesPermitted none			The switching function sends the Diverted event only to the divertingDevice.
5. The switching function clears D1C2.	Connection Cleared • droppedConnection D1C2 • releasingDevice D1 • localConnectionInfo null • cause normalClr • servicesPermitted none			

Table 5-30 Manual directed park (page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
6. The call is parked at device D3.		**Queued** • queuedConnection D3C1 • queue D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D1 • localConnectionInfo connected • cause park • servicesPermitted ClearConn, Hold, SendUserInfo	**Queued** • queuedConnection D3C1 • queue D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice D1 • localConnectionInfo queued • cause park • servicesPermitted SendUserInfo	
7. Device D1 goes blocked.	Failed • failedConnection D1C3 • failingDevice D1 • callingDevice NK • calledDevice NK • lastRedirectionDevice NS • localConnectionInfo fail • cause blocked • servicesPermitted ClearConn			
8. D1 goes on-hook	Connection Cleared • droppedConnection D1C3 • releasingDevice D1 • localConnectionInfo null • cause normalClr • servicesPermitted none			

Table 5-30 Manual directed park (page 2 of 2)

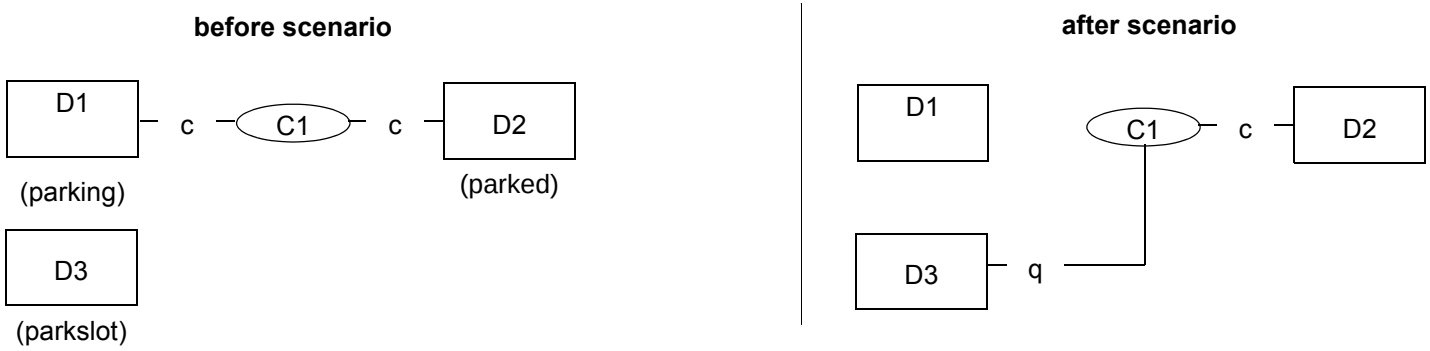
Remark:

The switching function does not change the calling, called parameters in the event flow.

ECMA TR/82 reports changing calling, called devices in the related scenario.

5.11.5 Manual system park

This scenario illustrates placing an established call on system park.



See “Succesful answer call” on page 5-19 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. D1 presses system park. The call has been diverted from D1, and placed on system hold.	Diverted • connection D1C1 • divertingDevice D1 • **newDestination** NK • callingDevice NK • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo null • cause park • servicesPermitted none		none	The switching function sends the Diverted event only to the divertingDevice. Proper newDestination parameter cannot be provided due to switching function limitation.

Table 5-31 Manual system park (page 1 of 2)

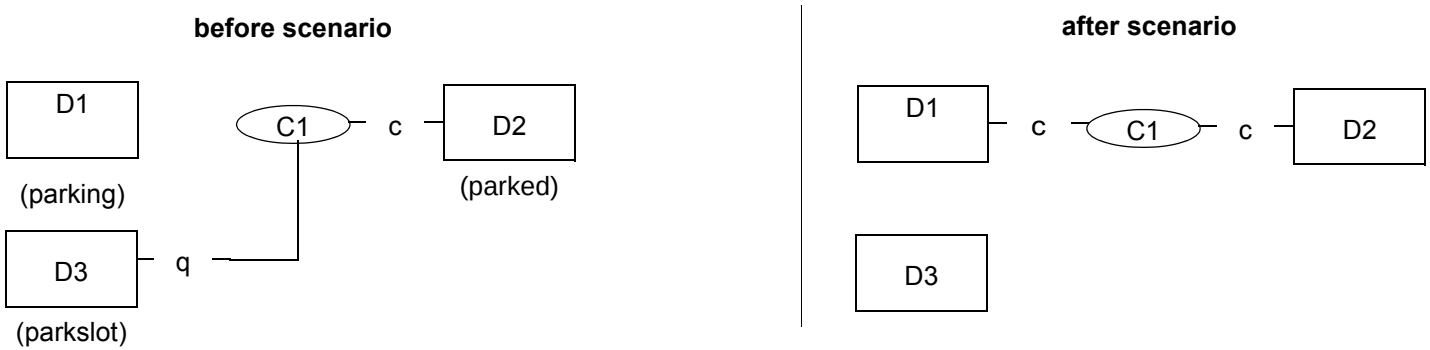
Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
2. D2 is parked to a parkslot.		**Queued** • queuedConnection D3C1 • queue D3 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause park • servicesPermitted ClearConn, Hold, SendUserInfo		A parkslot device cannot be monitored. That is why no event is reported for D3.

Table 5-31 Manual system park (page 2 of 2)

Remark:

None

5.11.6 Manual system unpark



See “Manual system park” for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. D2 hits the park key again: it invokes unpark.	Established • establishedConnection D1C1 • answeringDevice D1 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause park • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	Established • establishedConnection D1C1 • answeringDevice D1 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause park • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo		

5.12 Hold/Retrieving Scenarios

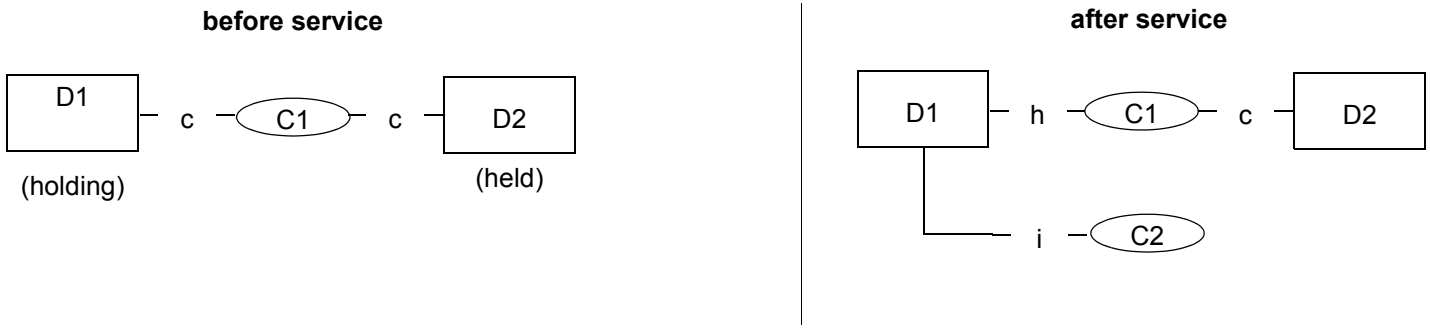
This section includes examples of successful Hold and Retrieve Call scenarios.

5.12.1 Hold Call

5.12.1.1 Hold Call, holding device is a Non-SIP device

This scenario illustrates the successful use of a Hold Call service. The service places an existing connection on hard-hold .

D1 is anate (analog telephone).



See “Succesful answer call” on page 5-19 for the event flow to get into the “before service” state.

Activity	Monitored Device D1		Monitored Device D2		Comments
1. Hold Call service is invoked on behalf of D1.	Hold Call Request • connToBeHeld	D1C1			
2. Acknowledgement.	Hold Call Response				
3. Connection placed on hold.	Held • heldConnection • holdingDevice • localConnectionInfo • cause • servicesPermitted	 D1C1 D1 held consultation SendUserInfo	Held • heldConnection • holdingDevice • localConnectionInfo • cause • servicesPermitted	 D1C1 D1 connected consultation ClearConn, SendUserInfo	
4. D1 stays offhook.	Service Initiated • initiatedConnection • initiatingDevice • localConnectionInfo • cause • servicesPermitted	 D1C2 D1 initiated consultation ClearConn, DialDgt, Reconnect			

Table 5-32 Hold Call

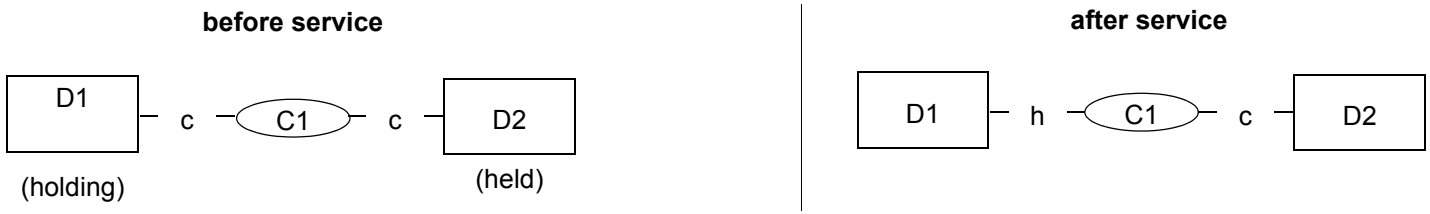
Remark:

The manual case is similar to the described event flow.

5.12.1.2 Hold Call, holding device is a SIP device

This scenario illustrates the successful use of a Hold Call service. The service places an existing connection on hard-hold .

D1 is a SIP device.



See “Succesful answer call” on page 5-19 for the event flow to get into the “before service” state.

Activity	Monitored Device D1		Monitored Device D2		Comments
1. Hold Call service is invoked on behalf of D1.	Hold Call Request • connToBeHeld	D1C1			
2. Acknowledgement.	Hold Call Response				
3. Connection placed on hold.	Held • heldConnection • holdingDevice • localConnectionInfo • cause • servicesPermitted	 D1C1 D1 hold normal none	Held • heldConnection • holdingDevice • localConnectionInfo • cause • servicesPermitted	 D1C1 D1 connected normal ClearConn, Hold, SendUserInfo	

Table 5-33 Hold Call, SIP device is holding

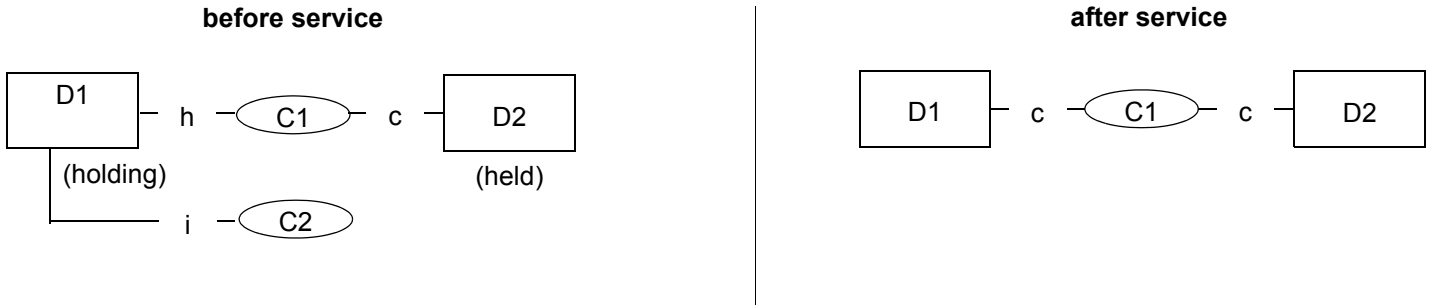
Remark:

The manual case is similar to the described event flow.

5.12.2 Retrieve Call

This service reconnects to a call that has previously been placed on hard-hold.

D1 is anate (analog telephone).



See “Hold Call” on page 5-64 for the event flow to get into the “before service” state

Activity	Monitored Device D1	Monitored Device D2	Comments
1. Retrieve Call service is invoked on behalf of D1.	Retrieve Call Request • heldConnection D1C1		
2. Acknowledgement.	Retrive Call Result Response		
3. Device D1 is cleared from the active call.	Connection Cleared • droppedConnection D1C2 • releasingDevice D1 • localConnectionInfo null • cause normalClr • servicesPermitted none		

Table 5-34 Retrieve Call (page 1 of 2)

Activity	Monitored Device D1		Monitored Device D2		Comments
4. Device D2 is connected back into the previously held call.	Retrieved • retrievedConnection • Retrieving • localConnectionInfo • cause • servicesPermitted	 D1C1 D1 connected normal ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	Retrieved • retrievedConnection • Retrieving • localConnectionInfo • cause • servicesPermitted	 D1C1 D1 connected normal ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	

Table 5-34 Retrieve Call (page 2 of 2)

Remark:

The manual case is similar to the described event flow.

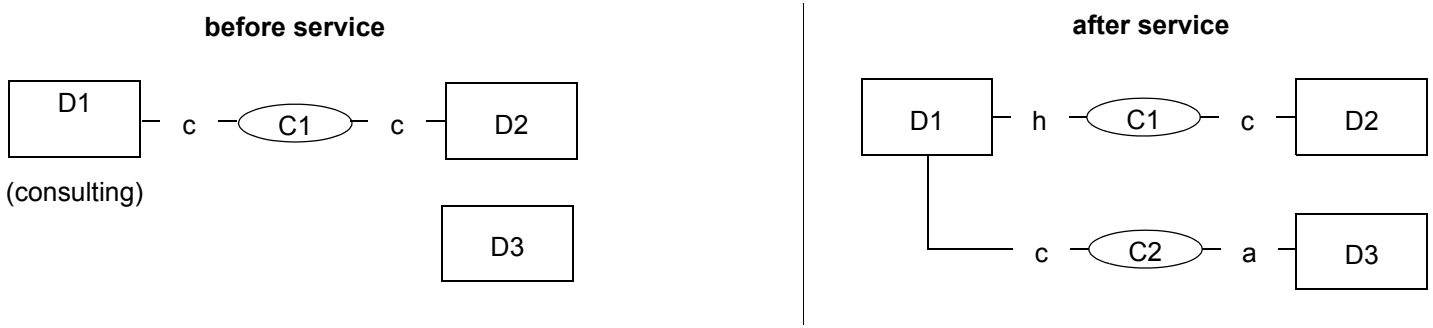
5.13 Consultation Call Scenarios

This section illustrates examples of successful Consultation Call, Reconnect Call and Alternate Call initiated by CSTA services.

5.13.1 Successful consultation Call

5.13.1.1 Consulting party is a Non-SIP device

This service places an existing active call at a device on hold and initiates a new call from the same device.



See “Succesful answer call” on page 5-19 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Consultation Call service is invoked on behalf of D1.	Consultation Call Request • connToBeHeld D1C1 • consultedDevice: D3			
2. Acknowledgement.	Consultation Call Response • consultedConnection D1C2			
3. Connection placed on hold.	Held • heldConnection D1C1 • holdingDevice D1 • localConnectionInfo held • cause consultation • servicesPermitted SendUserInfot	Held • heldConnection D1C1 • holdingDevice D1 • localConnectionInfo connected • cause consultation • servicesPermitted ClearConn, SendUserInfo		
4. D1 begins to dial.	Service Initiated • initiatedConnection D1C2 • initiatingDevice D1 • localConnectionInfo initiated • cause consultation • servicesPermitted ClearConn, Reconn, DialDgt			

Table 5-35 Consultation Call (page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
5. **D1 completes dialling D3's number ("1234")**	Digits Dialled • diallingConnection D1C1 • diallingDevice D1 • diallingSequence "1234" • localConnectionInfo initiated • cause consultation			
	Originated • originatedConnection D1C2 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo connected • cause consultation • servicesPermitted ClearConn			
6. D3 starts ringing.	Delivered • connection D3C2 • alertingDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted CallBack, ClearConn, SendUserInfo, Transfer, Reconnect		Delivered • connection D3C2 • alertingDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo alert • cause normal • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo	

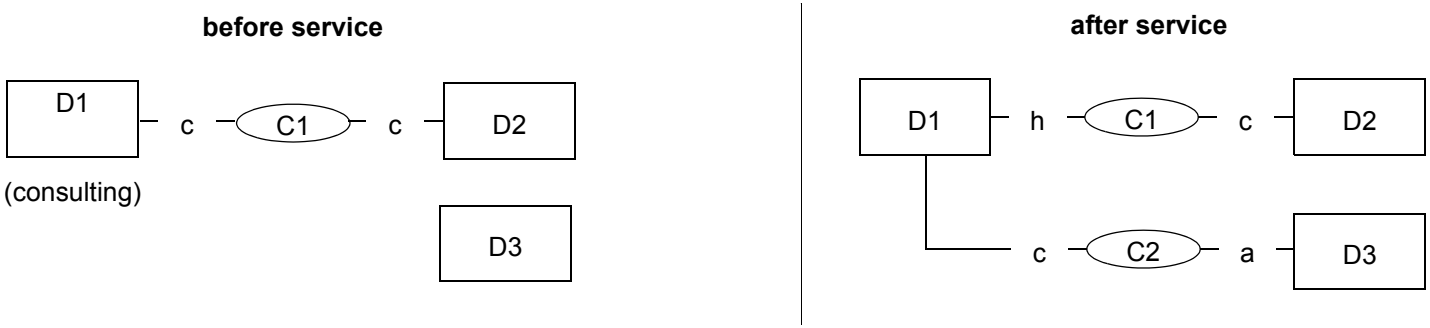
Table 5-35 Consultation Call (page 2 of 2)

Remark:

The manual case is similar to the described event flow.

5.13.1.2 Consulting party is a SIP device

This service places an existing active call at a device on hold and initiates a new call from the same device.



See “Succesful answer call” on page 5-19 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Consultation button is pressed on D1.	• connToBeHeld D1C1 • consultedDevice: D3			
2. Acknowledgement.	Consultation Call Response • consultedConnection D1C2			
3. Connection placed on hold.	Held • heldConnection D1C1 • holdingDevice D1 • localConnectionInfo held • cause normal • servicesPermitted none	Held • heldConnection D1C1 • holdingDevice D1 • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Hold, SendUserInfo		
4. D1 begins to dial.	Service Initiated • initiatedConnection D1C2 • initiatingDevice D1 • localConnectionInfo initiated • cause normal • servicesPermitted ClearConn, (from HP4k V6 only!)			

Table 5-36 Consultation Call (page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
5. **D1 completes dialling D3's number ("1234")**	Digits Dialled • diallingConnection D1C1 • diallingDevice D1 • diallingSequence "1234" • localConnectionInfo initiated • cause consultation			
	Originated • originatedConnection D1C2 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn (from HP4k V6 only!)			
6. D3 starts ringing.	Delivered • connection D3C2 • alertingDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn. (from HP4k V6 only!)		Delivered • connection D3C2 • alertingDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo alert • cause normal • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo	

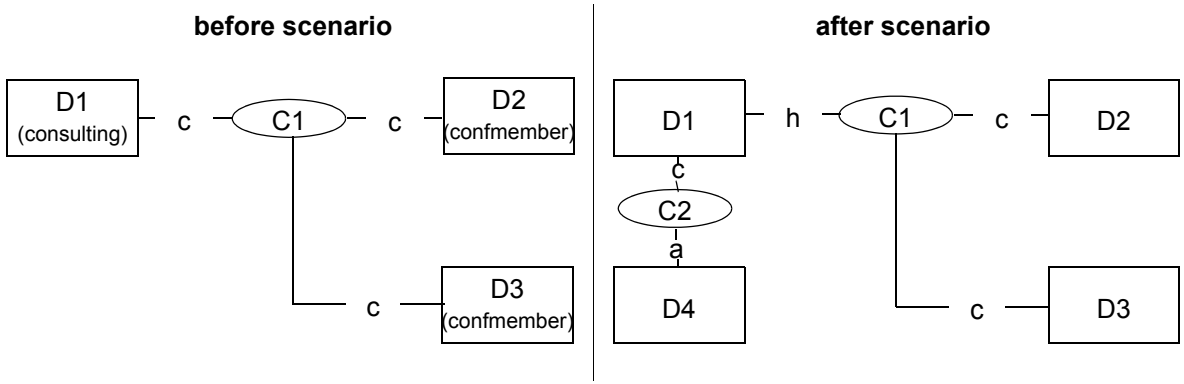
Table 5-36 Consultation Call (page 2 of 2)

Remark:

The manual case is similar to the described event flow.

5.13.2 Consulting out of a conference

This scenario illustrates the case when a conference member holds the conference. D1 is a Non-SIP device.



Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Device D1 hits the consultation key.	Held • heldConnection D1C1 • holdingDevice D1 • localConnectionInfo hold • cause consultation • servicesPermitted none			Note that there is no Held event for devices D2 and D3. This is a switching function limitation.
2. Since device D1 still off-hook it can dial.	Service Initiated • initiatedConnection D1C2 • initiatingDevice D1 • localConnectionInfo initiated • cause consultation • servicesPermitted ClearConn, DialDgt, Reconn			

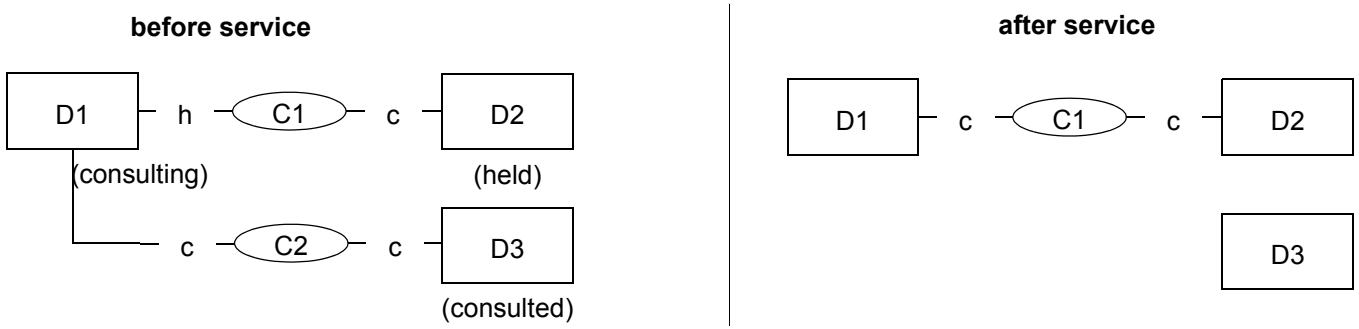
Table 5-37 Conference

Remark:

Events for D4 are not shown above, because they are not relevant in this context.

5.13.3 Reconnect Call

This service clears an existing connection and then retrieves a previously held connection at the same device.



See “Successful consultation Call” on page 5-68 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Retrieve Call service is invoked on behalf of D1.	Reconnect Call Request • heldConnection D1C1 • activeConnection D1C2			
2. Acknowledgement.	Reconnect Call Response			
3. Device D1 is cleared from the active call.	Connection Cleared • droppedConnection D1C2 • releasingDevice D1 • localConnectionInfo null • cause normalClr • servicesPermitted none		Connection Cleared • droppedConnection D1C2 • releasingDevice D1 • localConnectionInfo connected • cause normalClr • servicesPermitted ClearConn	

Table 5-38 Reconnect Call (page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
4. Device D1 is connected back to the previously held call.	Retrieved • retrievedConnection D1C1 • Retrieving D1 • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	Retrieved • retrievedConnection D1C1 • Retrieving D1 • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo		
5. As a result of D1C2 clearing, remining device D3 goes blocked.			Failed • failedConnection D3C2 • failingDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo fail • cause blocked • servicesPermitted ClearConn	
6. As a result of D1C2 clearing, D3C2 is also cleared.			Connection Cleared • droppedConnection D3C2 • releasingDevice D3 • localConnectionInfo null • cause normalClr • servicesPermitted none	

Table 5-38 Reconnect Call (page 2 of 2)

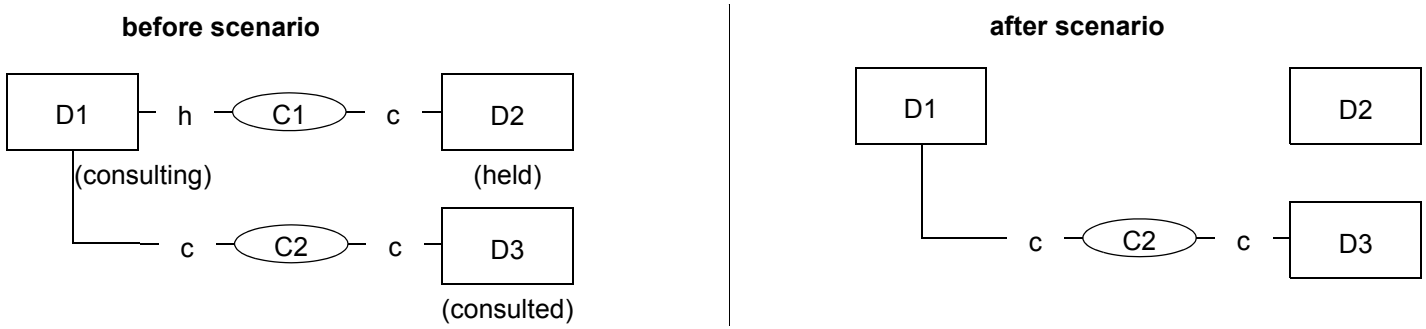
Remark:

The manual case is similar to the described event flow.

5.13.4 Held Party Releases

5.13.4.1 Consulting device is a Non-SIP device

This scenario illustrates the situation when the held party in a consultation releases the call.



See “Successful consultation Call” on page 5-68 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Device D2 is cleared from the held call.	Connection Cleared • droppedConnection D2C1 • releasingDevice D2 • localConnectionInfo held • cause normalClr • servicesPermitted none	Connection Cleared • droppedConnection D2C1 • releasingDevice D2 • localConnectionInfo null • cause normalClr • servicesPermitted none	none	
2. Device D1 is cleared from the held call.	Connection Cleared • droppedConnection D1C1 • releasingDevice D1 • localConnectionInfo null • cause normalClr • servicesPermitted none			
3. Call information is provided for device D1	CallInformation • ConnectionId D1C2 • Subject deviceId: D1 • servicesPermitted ClearConn, ConsultationCall, Hold, SST, GenDgt, GenTelTones, SendUserInfo			CallInformation event is generated to provide the changed permitted services.

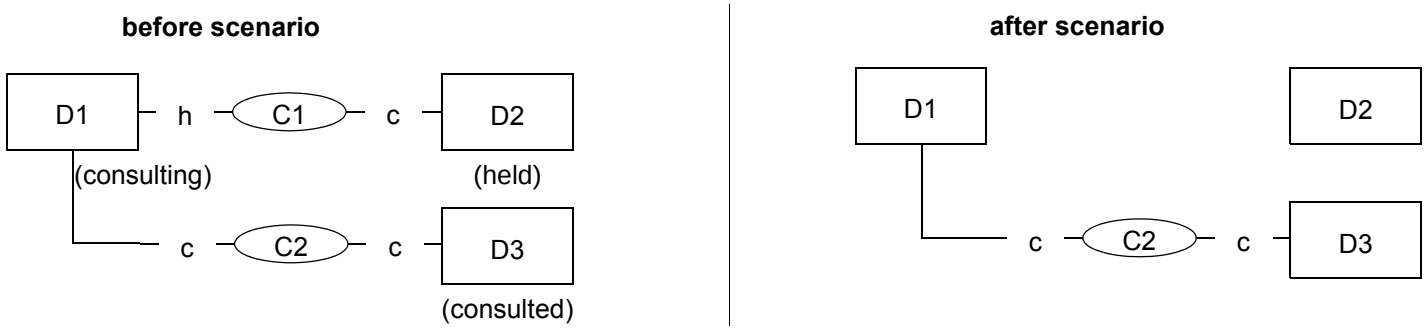
Table 5-39 Held Party Releases

Remark:

None

5.13.4.2 Consulting device is a SIP device

This scenario illustrates the situation when the held party in a consultation releases the call.



See “Successful consultation Call” on page 5-68 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Device D2 is cleared from the held call.	Failed • failedConnection D1C1 • failingDevice D1 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo fail • cause blocked • servicesPermitted ClearConn (from HP4k V6 only!)	Connection Cleared • droppedConnection D2C1 • releasingDevice D2 • localConnectionInfo null • cause normalClr • servicesPermitted none	none	

Table 5-40 Held Party Releases (page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
2. Device D1 is cleared from the held call.	Connection Cleared • droppedConnection D1C1 • releasingDevice D1 • localConnectionInfo null • cause normalClr • servicesPermitted none			

Table 5-40 Held Party Releases (page 2 of 2)

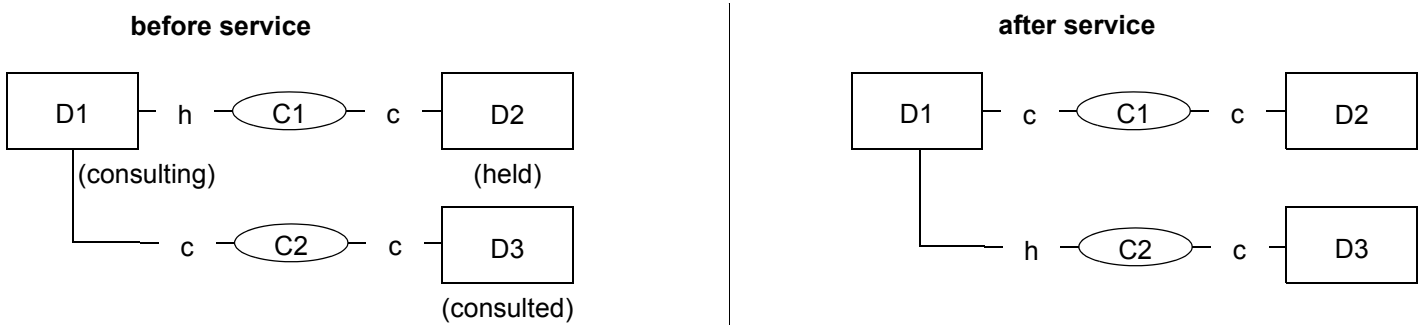
Remark:

None

5.13.5 Alternate Call

5.13.5.1 Consulting party is a Non-SIP device

This service places an existing active call on hold and then retrieves a previously held call at the same device. The effect of this service is to swap the device's active and held calls.



See "Successful consultation Call" on page 5-68 for the event flow to get into the "before service" state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Alternate Call service is invoked on behalf of D1.	Alternate Call Request • heldConnection D1C1 • activeConnection D1C2			
2. Acknowledgement.	Alternate Call Response			
3. Connection D1C2 is placed on hold in the active call.	Held • heldConnection D1C2 • holdingDevice D1 • localConnectionInfo held • cause alternate • servicesPermitted SendUserInfo		Held • heldConnection D1C2 • holdingDevice D1 • localConnectionInfo connected • cause alternate • servicesPermitted SendUserInfo, Hold, ClearConn	
4. Device D1 is connected back to the previously held call.	Retrieved • retrievedConnection D1C1 • Retrieving D1 • localConnectionInfo connected • cause alternate • servicesPermitted ClearConn, AlternateCall, ConferenceCall, GenTelTones, SendUserInfo	Retrieved • retrievedConnection D1C1 • Retrieving D1 • localConnectionInfo connected • cause alternate • servicesPermitted ClearConn, Consult, Hold, GenDgt, GenTelTones, SendUserInfo		

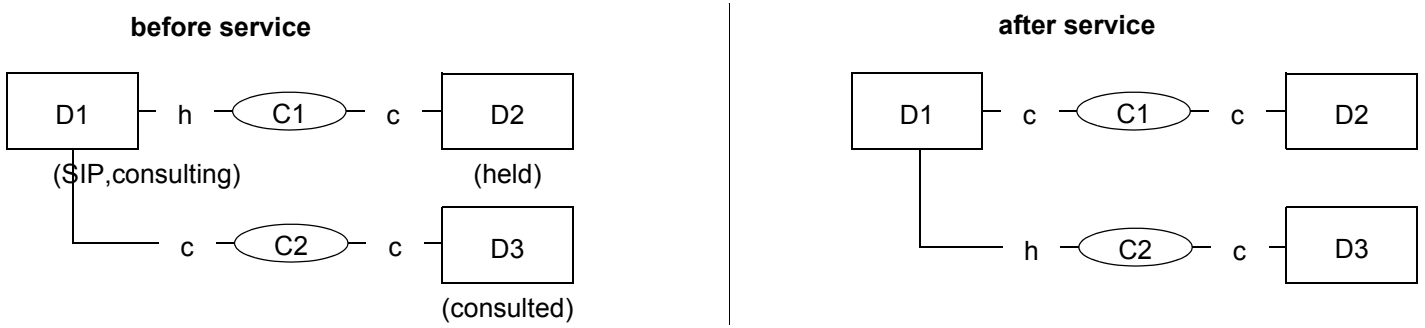
Table 5-41 Alternate Call

Remark:

The manual case is similar to the described event flow.

5.13.5.2 Consulting party is a SIP device

Consulting SIP party places an existing active call on hold and then retrieves a previously held call at the same device. The effect of this feature is to swap the device's active and held calls.



See “Successful consultation Call” on page 5-68 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. SIP presses alternate: Connection D1C2 is placed on hold in the active call.	Held • heldConnection D1C2 • holdingDevice D1 • localConnectionInfo hold • cause normal • servicesPermitted SendUserInfo		Held • heldConnection D1C2 • holdingDevice D1 • localConnectionInfo connected • cause normal • servicesPermitted SendUserInfo, Hold, ClearConn	
2. Device D1 is connected back to the previously held call.	Retrieved • retrievedConnection D1C1 • Retrieving D1 • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, SingleStepTransfer (from HP4k V6 only!)	Retrieved • retrievedConnection D1C1 • Retrieving D1 • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Consult, Hold, GenDgt, GenTelTones, SendUserInfo		

Table 5-42 Alternate Call on SIP device

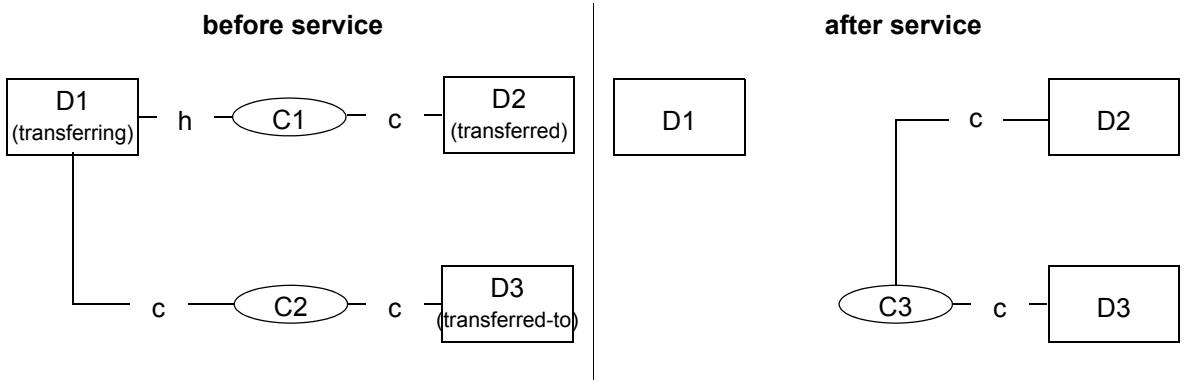
Remark:

5.14 Transfer Call Scenarios

5.14.1 Screened Transfer (with local view in Transferred event)

5.14.1.1 Transferring party is a Non-SIP device

This service transfers a held party to a consulted party.



See “Successful consultation Call” on page 5-68 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Transfer Call service is invoked on behalf of device D1.	Transfer Call Request • heldConnection D1C1 • activeConnection D1C2			
2. Acknowledgement.	Transfer Call Response • transferredConnection D3C3			

Table 5-43 Screened Transfer (page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
3. Calls between D1, D2 and D1, D3 are released. The connections between D2, D1 and D3, D1 are replaced with a single connection between D2 and D3.	Transferred • primaryOldCall D1C1 • secondaryOldCall D1C2 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D2C3) / (D2C1) 2. new / old (D3C3) / (D3C2) • localConnectionInfo null • cause Transfer • servicesPermitted none	Transferred • primaryOldCall D2C1 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D2C3) / (D2C1) 2. new (D3C3) • localConnectionInfo connected • cause Transfer • servicesPermitted ClearConn, Consult, Hold, SST, GenDg, GenTelTone, SendUserInfo	Transferred • primaryOldCall D3C2 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D3C3) / (D3C2) 2. new (D2C3) • localConnectionInfo connected • cause Transfer • servicesPermitted ClearConn, Consult, Hold, SST, GenDg, GenTelTone, SendUserInfo	The CSTA Transferred event Local View modeling option is provided by the switching function. This means that the primary old call parameters in the Transferred events represent a device oriented view.

Table 5-43 Screened Transfer (page 2 of 2)

Remark:

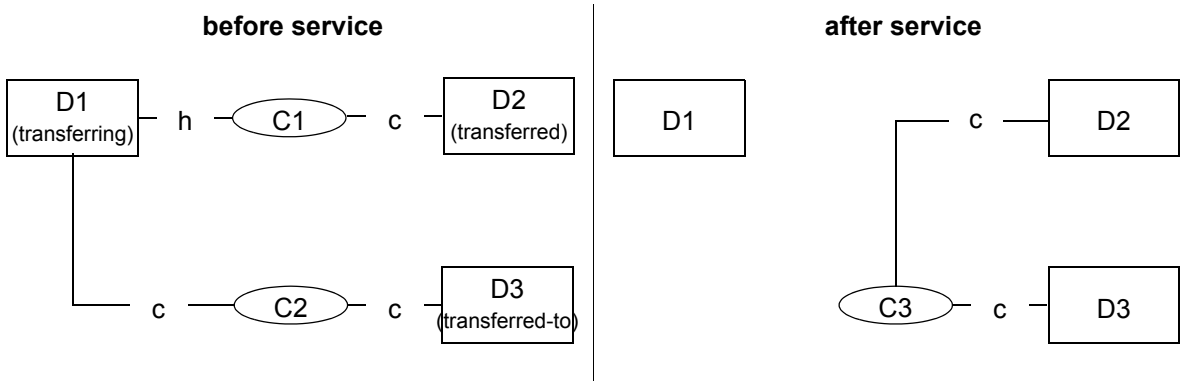
The manual case is similar to the described event flow.

In case of a transfer a new call Id (C3) will be generated that means the called device (D3) of this new call Id will not be the same as it was (D2) in the old call (C2).

For more information about Transferred local/global view, see ECMA TR/82 Scenarios for Computer Supported Telecommunication Applications (CSTA) Phase III, (December 2000) document on page 65.

5.14.1.2 Transferring party is a SIP device

This service transfers a held party to a consulted party.



See “Successful consultation Call” on page 5-68 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. SIP presses “Join”. Calls between D1, D2 and D1, D3 are released. The connections between D2, D1 and D3, D1 are replaced with a single connection between D2 and D3.	Transferred • primaryOldCall D1C1 • secondaryOldCall D1C2 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D2C3) / (D2C1) 2. new / old (D3C3) / (D3C2) • localConnectionInfo null • cause Transfer • servicesPermitted none	Transferred • primaryOldCall D2C1 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D2C3) / (D2C1) 2. new (D3C3) • localConnectionInfo connected • cause Transfer • servicesPermitted ClearConn, Consult, Hold, SST, GenDg, GenTelTone, SendUserInfo	Transferred • primaryOldCall D3C2 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D3C3) / (D3C2) 2. new (D2C3) • localConnectionInfo connected • cause Transfer • servicesPermitted ClearConn, Consult, Hold, SST, GenDg, GenTelTone, SendUserInfo	The CSTA Transferred event Local View modeling option is provided by the switching function. This means that the primary old call parameters in the Transferred events represent a device oriented view.

Table 5-44 Screened Transfer, transfering party is SIP(page 1 of 2)

Activity	Monitored Device D1		Monitored Device D2		Monitored Device D3		Comments
4. D1 goes in blocked state	Failed						
	• failedConnection	D1C1					
	• failingDevice	D1					
	• callingDevice	D1					
	• calledDevice	D2					
	• lastRedirectionDevice	NS					
	• localConnectionInfo	fail					
	• cause	blocked					
	• servicesPermitted	ClearConn					
	Connection Cleared						
	• droppedConnection	D1C1					
	• releasingDevice	D1					
	• localConnectionInfo	null					
	• cause	normalClr					
	• servicesPermitted	none					

Table 5-44 Screened Transfer, transfering party is SIP(page 2 of 2)

Remark:

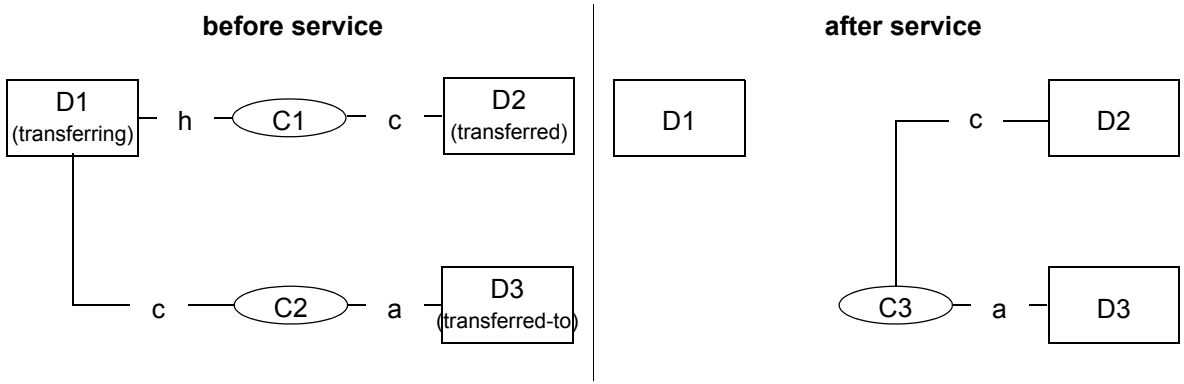
In case of a transfer a new call Id (C3) will be generated that means the called device (D3) of this new call Id will not be the same as it was (D2) in the old call (C2).

For more information about Transferred local/global view, see ECMA TR/82 Scenarios for Computer Supported Telecommunication Applications (CSTA) Phase III, (December 2000) document on page 65.

5.14.2 Blind Transfer (with local view in Transferred event)

5.14.2.1 Transferring device is a Non-SIP device

This service transfers a held party to a consulted party. The transfer service request is issued before the consulted device connects into the new call.



See “Successful consultation Call” on page 5-68 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Transfer Call service is invoked on behalf of device D1.	Transfer Call Request • heldConnection D1C1 • activeConnection D1C2			
2. Acknowledgement.	Transfer Call Response • transferredConnection D3C3			

Table 5-45 Blind Transfer (page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
3. Calls between D1, D2 and D1, D3 are released. The connections between D2, D1 and D3, D1 are replaced with a single connection between D2 and D3.	Transferred • primaryOldCall D1C1 • secondaryOldCall D1C2 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D2C3) / (D2C1) 2. new / old (D3C3) / (D3C2) • localConnectionInfo null • cause Transfer • servicesPermitted none	Transferred • primaryOldCall D2C1 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D2C3) / (D2C1) 2. new (D3C3) • localConnectionInfo connected • cause Transfer • servicesPermitted ClearConn, SendUserInfo	Transferred • primaryOldCall D3C2 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D3C3) / (D3C2) 2. new (D2C3) • localConnectionInfo alerting • cause Transfer • servicesPermitted Answer, ClearConn, SendUserInfo	The CSTA Transferred event Local View modeling option is provided by the switching function. This means that the primary old call parameters in the Transferred event represent a device oriented view.

Table 5-45 Blind Transfer (page 2 of 2)

Remark:

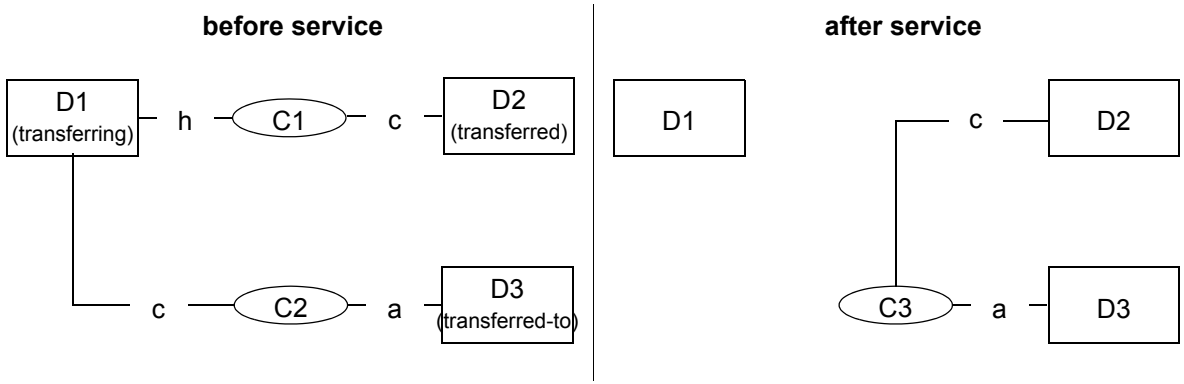
The manual case is similar to the described event flow.

In case of a transfer a new call Id (C3) will be generated that means the called device (D3) of this new call Id will not be the same as it was (D2) in the old call (C2).

For more information about Transferred local/global view, see ECMA TR/82 Scenarios for Computer Supported Telecommunication Applications (CSTA) Phase III, (December 2000) document on page 65.

5.14.2.2 Transferring device is a SIP device

The SIP device transfers a held party to a consulted party. The transfer is issued before the consulted device connects into the new call.



See “Successful consultation Call” on page 5-68 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. SIP device presses “Join”. Calls between D1, D2 and D1, D3 are released. The connections between D2, D1 and D3, D1 are replaced with a single connection between D2 and D3.	Transferred • primaryOldCall D1C1 • secondaryOldCall D1C2 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D2C3) / (D2C1) 2. new / old (D3C3) / (D3C2) • localConnectionInfo null • cause Transfer • servicesPermitted none	Transferred • primaryOldCall D2C1 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D2C3) / (D2C1) 2. new (D3C3) • localConnectionInfo connected • cause Transfer • servicesPermitted ClearConn, SendUserInfo	Transferred • primaryOldCall D3C2 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D3C3) / (D3C2) 2. new (D2C3) • localConnectionInfo alerting • cause Transfer • servicesPermitted Answer, ClearConn, SendUserInfo	The CSTA Transferred event Local View modeling option is provided by the switching function. This means that the primary old call parameters in the Transferred event represent a device oriented view.

Table 5-46 Blind Transfer on SIP device(page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
4.	Failed • failedConnection D1C1 • failingDevice D1 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo fail • cause blocked • servicesPermitted ClearConn			
	Connection Cleared • droppedConnection D1C1 • releasingDevice D1 • localConnectionInfo null • cause normalClr • servicesPermitted none			

Table 5-46 Blind Transfer on SIP device(page 2 of 2)

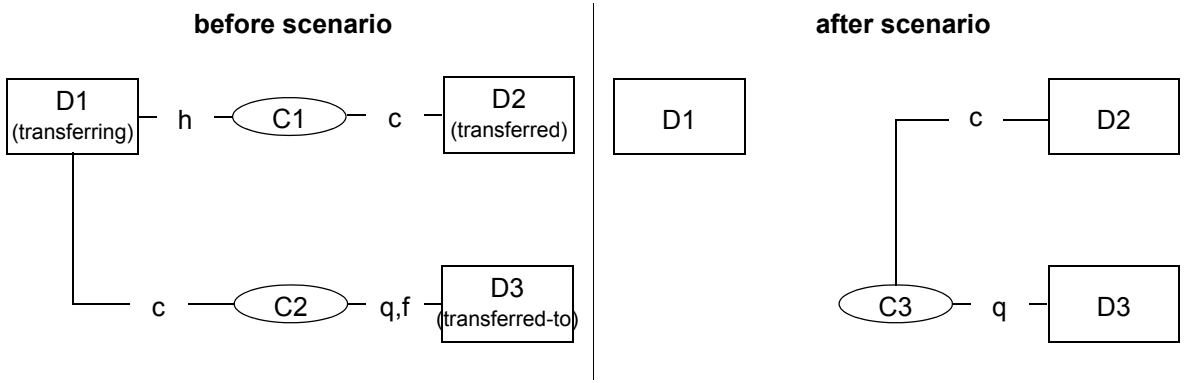
Remark:

In case of a transfer a new call Id (C3) will be generated that means the called device (D3) of this new call Id will not be the same as it was (D2) in the old call (C2).

For more information about Transferred local/global view, see ECMA TR/82 Scenarios for Computer Supported Telecommunication Applications (CSTA) Phase III, (December 2000) document on page 65.

5.14.3 Transfer to a busy station (with local view in Transferred event)

This service transfers a held party to a busy party. The transferred party camps-on to the transferred-to party.



Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. The switch automaticaly clears the failed connection.	Connection Cleared • droppedConnection D3C2 • releasingDevice D3 • localConnectionInfo connected • cause normalClr • servicesPermitted ClearConn		Connection Cleared • droppedConnection D3C2 • releasingDevice D3 • localConnectionInfo null • cause normalClr • servicesPermitted none	
2. Transfer Call service is invoked on behalf of device D1.	Transfer Call Request • heldConnection D1C1 • activeConnection D1C2			
3. Acknowledgement.	Transfer Call Response • transferredConnection D3C3			

Table 5-47 Transfer to a busy station (page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
4. Calls between D1, D2 and D1, D3 are released. The connections between D2, D1 and D3, D1 are replaced with a single connection between D2 and D3.	Transferred • primaryOldCall D1C1 • secondaryOldCall D2C2 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D2C3) / (D2C1) 2. new / old (D3C3) / (D3C2) • localConnectionInfo null • cause campOn • servicesPermitted none	Transferred • primaryOldCall D2C1 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D2C3) / (D2C1) 2. new (D3C3) • localConnectionInfo connected • cause normal • servicesPermitted CallBack, ClearConn, SendUserInfo	Transferred • primaryOldCall D3C2 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new / old (D3C3) / (D3C2) 2. new (D2C3) • localConnectionInfo fail • cause normal • servicesPermitted SendUserInfo	The CSTA Transferred event Local View modeling option is provided by the switching function. This means that the primary old call parameters in the Transferred event represent a device oriented view.
5. D2 camps on D3.		Queued • queuedConnection D3C3 • queue D3 • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo connected • cause campOn • servicesPermitted CallBack, ClearConn, SendUserInfo	Queued • queuedConnection D3C3 • queue D3 • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo queued • cause campOn • servicesPermitted SendUserInfo	

Table 5-47 Transfer to a busy station (page 2 of 2)

Remark:

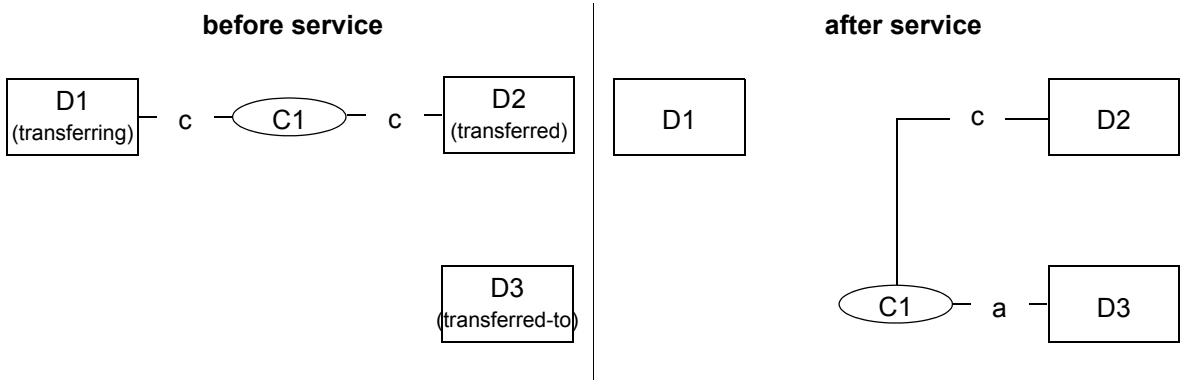
The manual case is similar to the described event flow.

In case of a transfer a new call Id (C3) will be generated that means the called device (D3) of this new call Id will not be the same as it was (D2) in the old call (C2).

For more information about Transferred local/global view, see ECMA TR/82 Scenarios for Computer Supported Telecommunication Applications (CSTA) Phase III, (December 2000) document on page 65.

5.14.4 Single Step Transfer (with local view in Transferred event)

This service transfers a device in one step.



See “Succesful answer call” on page 5-19 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Single Step Transfer Call service is invoked on behalf of device D1.	Single Step Transfer Call Request • activeConnection D1C1 • transferredTo D3			
2. Acknowledgement.	Single Step Transfer Call Response • transferredConnection D3C1			

Table 5-48 Single Step Transfer (Devices) (page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
3. The call between D1 and D2 is replaced with an alerting call between D2 and D3.	Transferred • primaryOldCall D1C1 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new D2C1 2. new D3C1 • localConnectionInfo null • cause SST • servicesPermitted none	Transferred • primaryOldCall D2C1 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new: D2C1 2. new D3C1 • localConnectionInfo connected • cause SST • servicesPermitted ClearConn, SendUserInfo		The CSTA Transferred event Local View modeling option is provided by the switching function. This means that the primary old call parameters in the Transferred event represent a device oriented view. Note that the switching function will not provide a new call id.
4. The call alerts device D3.		Delivered • connection D3C1 • alertingDevice D3 • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo connected • cause SST • servicesPermitted ClearConn, SendUserInfo	Delivered • connection D3C1 • alertingDevice D3 • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo alerting • cause SST • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo	This event reflects the connection state change at D3C1.

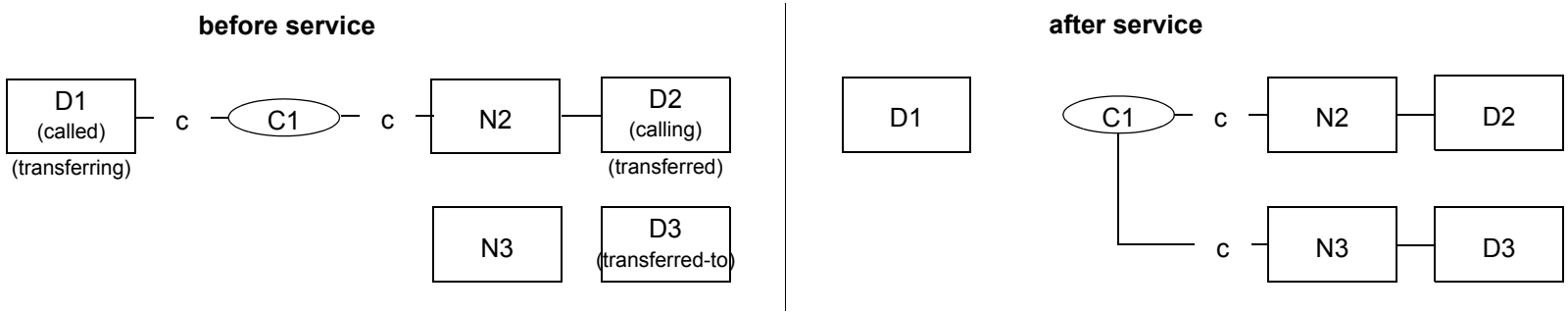
Table 5-48 Single Step Transfer (Devices) (page 2 of 2)

Remark:

The swithing function does not allocate a new callID in case of a Single Step Transfer.

5.14.5 Single Step Transfer between network interface devices (with local view in Transferred event)

This scenario illustrates a successful Single Step Transfer Service. The transferred and transferred-to devices are network interface devices (NIDs) .



See “External incoming calls” on page 5-30 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device N2	Monitored Device N3	Comments
1. Single Step Transfer Call service is invoked on behalf of device D1.	Single Step Transfer Call Request • activeConnection D1C1 • transferredTo D3			
2. Acknoledgement .	Transfer Call Response • transferredConn D3C1			

Table 5-49 Single Step Transfer (Trunk to Trunk) (page 1 of 3)

Activity	Monitored Device D1	Monitored Device N2	Monitored Device N3	Comments
3. The network is reached again.	Network Reached • outboundConnection N3C1 • networkInterfaceUsed N3 • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted none	Network Reached • outboundConnection N3C1 • networkInterfaceUsed N3 • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, SendUserInfo • networkCallingDevice D2 • assocCallingDevice N2	Network Reached • outboundConnection N3C1 • networkInterfaceUsed N3 • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Deflect, SendUserInfo • networkCallingDevice D2 • assocCallingDevice N2	
4. Device D2 transfers.	Transferred • primaryOldCall D1C1 • • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new:assNID:endp (N2C1):N2:D2 2. new:assNID:endp (N3C1):N3:D3 • localConnectionInfo null • cause SST • servicesPermitted none	Transferred • primaryOldCall N2C1 • transferringDevice D1 • transferredToDevice D3 • transferredConnections 1. new/ old:assNID:endp (N2C1)/ (N2C1):N2:D2 2. new:assNID:endp (N3C1):N3:D3 • localConnectionInfo connected • cause SST • servicesPermitted ClearConn, SendUserInfo		The CSTA Transferred event Local View modeling option is provided by the switching function. This means that the primary old call parameters in the Transferred event represent a device oriented view. Note that the switching function will not provide a new call id.

Table 5-49 Single Step Transfer (Trunk to Trunk) (page 2 of 3)

Activity	Monitored Device D1	Monitored Device N2	Monitored Device N3	Comments
5. The call alerts device D3.		Delivered • connection N3C1 • alertingDevice D3 • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NS • origNID N2C1 • localConnectionInfo connected • cause SST • servicesPermitted ClearConn, SendUserInfo • networkCallingDevice D2 • assocCallingDevice N2 • assocCalledDevice N3	Delivered • connection N3C1 • alertingDevice D3 • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NS • origNID N2C1 • localConnectionInfo alert • cause SST • servicesPermitted ClearConn, Deflect, SendUserInfo • networkCallingDevice D2 • assocCallingDevice N2 • assocCalledDevice N3	This event reflects the connection state change at N3C1.

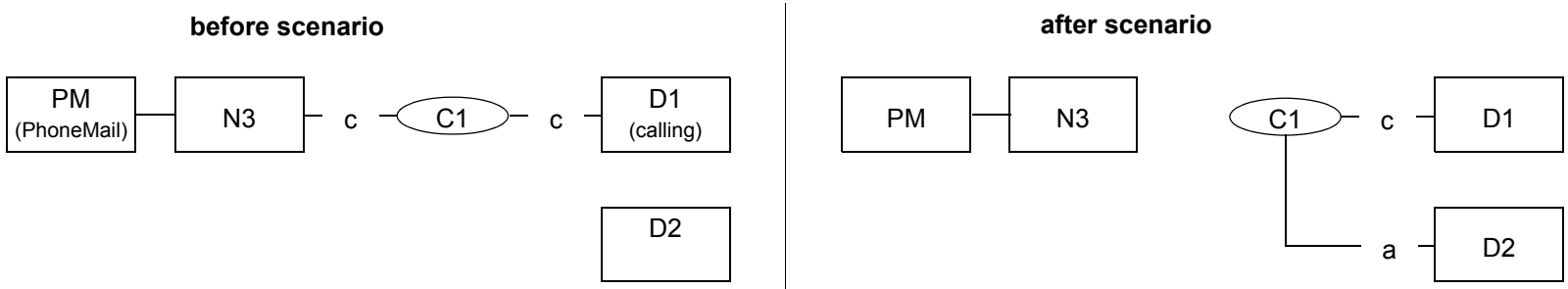
Table 5-49 Single Step Transfer (Trunk to Trunk) (page 3 of 3)

Remark:

The swithing function does not allocate a new callID in case of a Single Step Transfer.

5.14.6 Single Step Call Transfer, Phone Mail transfers

The scenario describes an event flow when a Phone Mail device transfers in one step.



See “Internal ACD call completed to Phone Mail agent” on page 5-117 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device N3	Monitored Device D2	Comments
1. The call between D1 and N3 is replaced with an alerting call between D1 and D2.	Transferred • primaryOldCall D1C1 • transferringDevice PM • transferredToDevice D2 • transferredConnections 1.new (D1C1) 2.new (D2C1) • localConnectionInfo connected • cause SST • servicesPermitted CallBack, ClearConn, SendUserInfo	Transferred • primaryOldCall N3C1 • transferringDevice PM • transferredToDevice D2 • transferredConnections 1.new (D1C1) 2.new (D2C1) • localConnectionInfo null • cause SST • servicesPermitted none		The CSTA Transferred event Local View modeling option is provided by the switching function. This means that the primary old call parameters in the Transferred event represent a device oriented view.
2. The call alerts D2.	Delivered • connection D2C1 • alertingDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause SST • servicesPermitted CallBack, ClearConn, SendUserInfo		Delivered • connection D2C1 • alertingDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo alerting • cause SST • servicesPermitted ClearConn, Answer, Deflect, SendUserInfo	These events reflect the connection state change at D2C1.

Table 5-50 Single Step Transfer from Phone Mail to Agent (page 1 of 2)

Activity	Monitored Device D1	Monitored Device N3	Monitored Device D2	Comments
3. Phone Mail goes into blocked state.		Failed • failedConnection N3C2 • failingDevice N3 • callingDevice NK • calledDevice NK • lastRedirectionDevice NS • localConnectionInfo fail • cause blocked • servicesPermitted ClearConn		
4. N3C3 is dropped.		Connection Cleared • droppedConnection N3C2 • releasingDevice N3 • localConnectionInfo null • cause normalClr • servicesPermitted none		

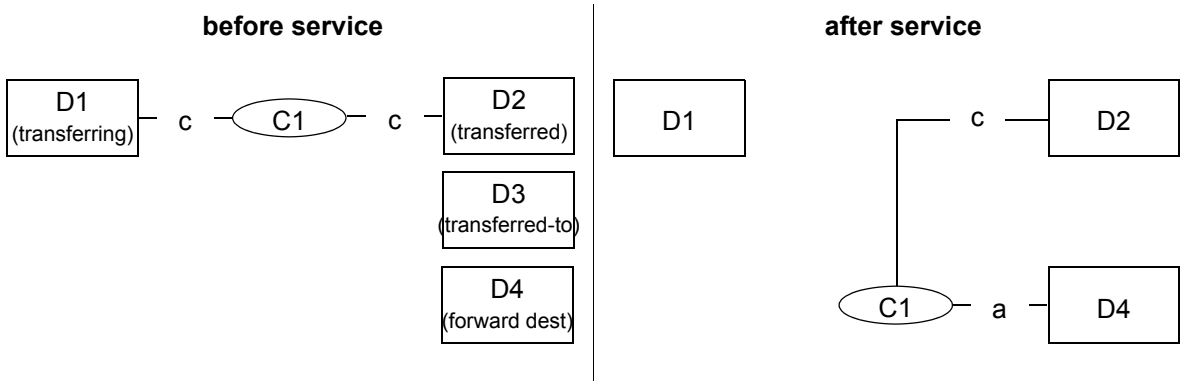
Table 5-50 Single Step Transfer from Phone Mail to Agent (page 2 of 2)

Remark:

None

5.14.7 Single Step Transfer to destination with call forward immediate handled in the switching subdomain (D3 is internal analogue or digital device)

This service transfers and forwards a connection in one step.



See “Succesful answer call” on page 5-19 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D4	Comments
1. Single Step Transfer Call service is invoked on behalf of device D1.	Single Step Transfer Call Request • activeConnection D1C1 • transferredTo D3			
2. Acknowledgement.	Single Step Transfer Call Response • transferredConnection D4C1			
3. The call between D1 and D2 is replaced with an alerting call between D2 and D3.	Transferred • primaryOldCall D1C1 • transferringDevice D1 • transferredToDevice D4 • transferredConnections 1. new D2C1 2. new D4C1 • localConnectionInfo null • cause SST • servicesPermitted none	Transferred • primaryOldCall D2C1 • transferringDevice D1 • transferredToDevice D4 • transferredConnections 1. new: D2C1 2. new D4C1 • localConnectionInfo connected • cause SST • servicesPermitted ClearConn, SendUserInfo		The CSTA Transferred event Local View modeling option is provided by the switching function. This means that the primary old call parameters in the Transferred event represent a device oriented view. Note that the switching function will not provide a new call id.

Table 5-51 Single Step Transfer with call forward (Devices) (page 1 of 2)

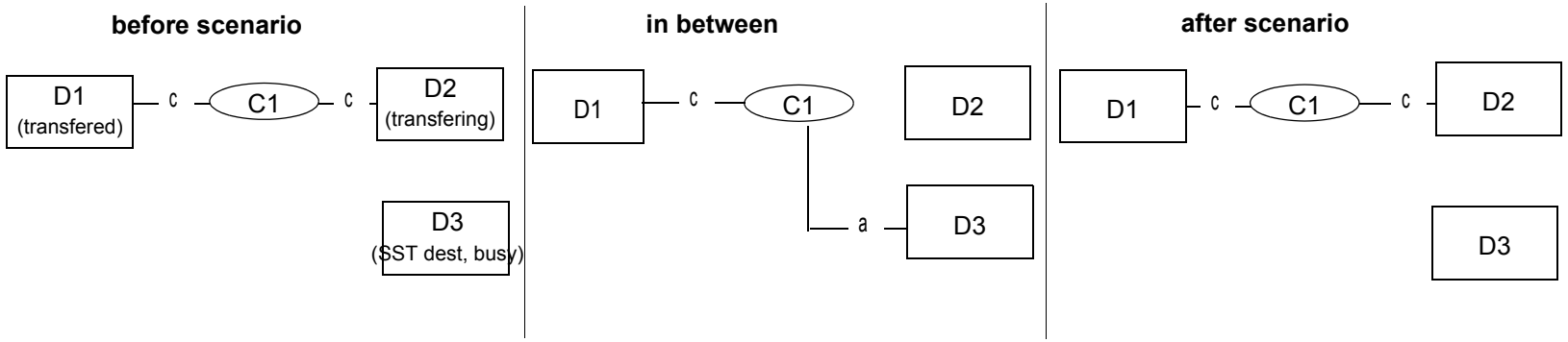
Activity	Monitored Device D1	Monitored Device D2	Monitored Device D4	Comments
4. The call alerts device D3.		Delivered • connection D4C1 • alertingDevice D4 • callingDevice D2 • calledDevice D4 • lastRedirectionDevice NS • localConnectionInfo connected • cause SST • servicesPermitted ClearConn, SendUserInfo	Delivered • connection D4C1 • alertingDevice D4 • callingDevice D2 • calledDevice D4 • lastRedirectionDevice NS • localConnectionInfo alerting • cause SST • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo	This event reflects the connection state change at D4C1.

Table 5-51 Single Step Transfer with call forward (Devices) (page 2 of 2)

Note:

Please note that the forward can only be followed by the change in the destination.

5.14.8 Single Step Transfer attempt to busy destination with offered mode activated.



See “Succesful answer call” on page 5-19 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Single Step Transfer Call service is invoked on behalf of device D1.	Single Step Transfer Call Request • activeConnection D2C1 • transferredTo D3			
2. Acknowledgement.	Single Step Transfer Call Response • transferredConnection D3C1			
3. The call between D1 and D2 is replaced with an alerting call between D2 and D3.	Transferred • primaryOldCall D1C1 • transferringDevice D2 • transferredToDevice D3 • transferredConnections 1. new D1C1 2. new D3C1 • localConnectionInfo connectedl • cause SST • servicesPermitted ClearConn, SendUserInfo	Transferred • primaryOldCall D2C1 • transferringDevice D2 • transferredToDevice D3 • transferredConnections 1. new: D1C1 2. new D3C1 • localConnectionInfo null • cause SST • servicesPermitted none		The CSTA Transferred event Local View modeling option is provided by the switching function. This means that the primary old call parameters in the Transferred event represent a device oriented view. Note that the switching function will not provide a new call id.

Table 5-52 Single Step Transfer attempt to busy destination with offered mode activated(page 1 of 3)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
4. The call is offered to D3.	Offered (optional) • connection D3C1 • offeredDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice D2 • localConnectionInfo connected • cause SST • servicesPermitted ClearConn, SendUserInfo		Offered • connection D3C1 • offeredDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice D2 • localConnectionInfo alerting • cause SST • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo	This event reflects the connection state change at D4C1.
5. Application accepts the call			Accept Cal lRequest • callToBeAccepted D3C1	
6. Acknowledged			Accept Call Response	
7. Destination is busy	Failed • failedConnection D3C1 • failingDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo connected • cause busy • servicesPermitted ClearConn		Failed • failedConnection D3C1 • failingDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo null • cause busy • servicesPermitted none	
8. Recall D2	Diverted (optional) • connection D3C1 • divertingDevice D3 • newDestination D2 • Calling D1 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo connected • cause recallBusy • servicesPermitted none		Diverted • connection D3C1 • divertingDevice D3 • newDestination D2 • Calling D1 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo null • cause recallBusy • servicesPermitted none	
9. Call reconnected to	Established • establishedConnection D2C1 • answerindDevice D2 • callingDevice D1 • calledDevice D2	Established • establishedConnection D2C1 • answerindDevice D2 • callingDevice D1 • calledDevice D2		

Table 5-52 Single Step Transfer attempt to busy destination with offered mode activated(page 2 of 3)

Activity	Monitored Device D1		Monitored Device D2		Monitored Device D3	Comments
	• lastRedirectionDevice	D3	• lastRedirectionDevice	D3		
	• localConnectionInfo	connected	• localConnectionInfo	connected		
	• cause	recallBusy	• cause	recallBusy		
	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDg, GenTelTone, SendUserInfo	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDg, GenTelTone, SendUserInfo		

Table 5-52 Single Step Transfer attempt to busy destination with offered mode activated(page 3 of 3)

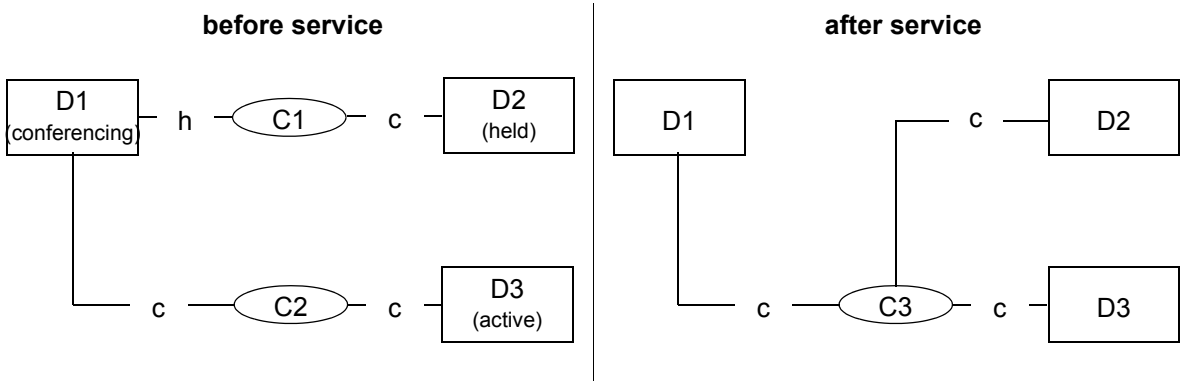
Note:This event flow can occur in cases where the Offered mode is activated on the destination and the ONS monitoring is activated. See Service Manual for details.

5.15 Conference Call Scenarios

5.15.1 Conference (with local view in Conferenced event)

5.15.1.1 Conference master is a Non-SIP device

This service provides a conference of an existing held call and another active call at a conferencing device. The two calls are merged into a single call at the conferencing device.



See “Successful consultation Call” on page 5-68 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Conference Call service is requested on behalf of device D1.	Conference Request • heldConnection D1C1 • activeConnection D1C2			
2. Acknowledgement.	Conference Response • conferencedConnection D1C3			
3. Conference estabilished.	Conferenced • primaryOldCall D1C1 • secondaryOldCall D1C2 • conferencingDevice D1 • Added D3 • conferenceConnections 1. new/old (D1C3)/(D1C1) 2. new/old (D1C3)/(D1C2) 3. new (D2C3) 4. new (D3C3) • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Consult, Hold, SendUserInfo	Conferenced • primaryOldCall D2C1 • conferencingDevice D1 • Added D3 • conferenceConnections 1. new/old (D2C3)/(D2C1) 2. new/old (D1C3)/(D1C1) 3. new (D3C3) • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Consult, Hold, SendUserInfo	Conferenced • primaryOldCall D3C2 • conferencingDevice D1 • Added D3 • conferenceConnections 1. new/old (D1C3)/(D1C2) 2. new/old (D3C3)/(D3C2) 3. new (D2C3) • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Consult, Hold, SendUserInfo	The addedParty specifies the device ID of the device, that belongs to the active (not held) call of the conference. Note that the primaryOldCall and the secondaryOldCall parameters follows the “local view” modeling option.

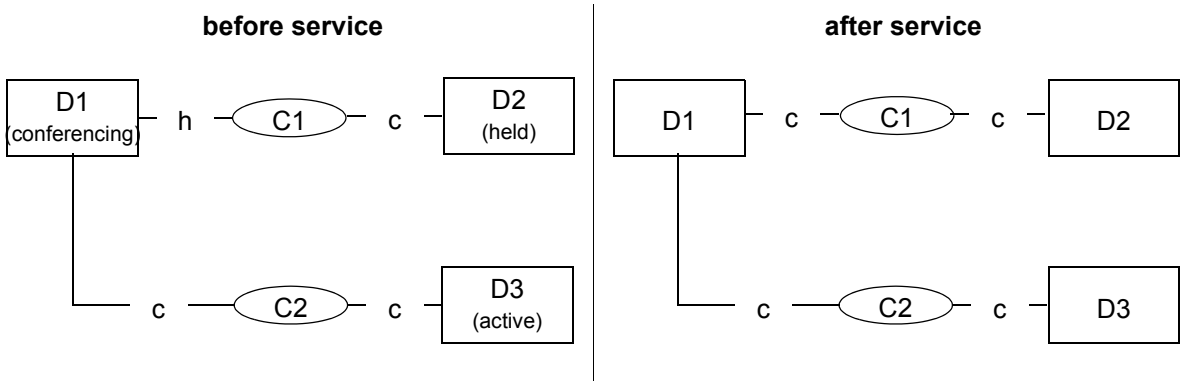
Table 5-53 Conference

Remark:

The manual case is similar to the described event flow.

5.15.1.2 Conference master is a SIP device

This service provides a conference of his existing 2 active calls at a conferencing device. The held call will be retrieved.



See “Successful consultation Call” on page 5-68 for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Conference button is preshed on device D1.	• heldConnection D1C1 • activeConnection D1C2			
2. Conference estabilished.	Retrieved • retrievedConnection D1C1 • Retrieving D1 • localConnectionInfo connecting • cause normal • servicesPermitted ClearConn, SingleStepTransfer (from HP4k V6 only!)	Retrieved • retrievedConnection D1C1 • Retrieving D1 • localConnectionInfo connecting • cause normal • servicesPermitted ClearConn, Consult, Hold, SingleStepTransfer, GenerateDigits, SendUserInfo		

Table 5-54 Conference

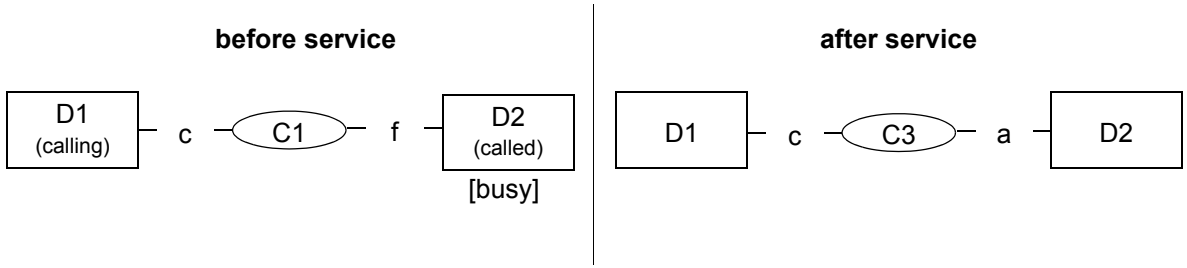
Remark:

SIP device can’t add more conference member.

5.16 Call Completion Scenarios

5.16.1 Call Back Call Related

This scenario illustrates the use of the Call Back Call Related service where the called device is busy.



See “Manually dialled call - called party is busy” on page 5-5 for the event flow to get into the “before service” state.

Activity	Monitored Device D1		Monitored Device D2		Comments
1. The busy connection is cleared immediately.	Connection Cleared		Connection Cleared		
	• droppedConnection	D2C1	• droppedConnection	D2C1	
	• releasingDevice	D2	• releasingDevice	D2	
	• localConnectionInfo	connected	• localConnectionInfo	null	
	• cause	normalClr	• cause	normalClr	
	• servicesPermitted	ClearConn	• servicesPermitted	none	
2. The Call Back Call Related service is invoked on behalf of device D1.	CallBack Request				
	• connection	D1C1			
3. Acknowledgement.	CallBack Response				
	• targetDevice	D2			

Table 5-55 Call Back Call Related (page 1 of 3)

Activity	Monitored Device D1	Monitored Device D2	Comments
4. Device D1 is blocked.	Failed • failedConnection D1C1 • failingDevice D1 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo fail • cause blocked • servicesPermitted ClearConn		
5. Device D1 clears its failed connection.	Connection Cleared • droppedConnection D1C1 • releasingDevice D1 • localConnectionInfo null • cause normalClr • servicesPermitted none		
6. Device D2 sometime later clears from its active call.		Connection Cleared • droppedConnection D2C2 • releasingDevice D2 • localConnectionInfo null • cause normalClr • servicesPermitted none	C2 is the active call of D2.
7. Since device D2 is now avaible, the CallBack is initiated from device D1. D1 is being prompted to go off-hook.	Service Initiated • initiatedConnection D1C3 • initiatingDevice D1 • localConnectionInfo initiated • cause CallBack • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo		The cause code of CallBack indicates that the device is being prompted to go off-hook.
8. The switching function reserves the CallBack destination (D2).		Failed • failedConnection D2C3 • failingDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo fail • cause blocked • servicesPermitted ClearConn	

Table 5-55 Call Back Call Related (page 2 of 3)

Activity	Monitored Device D1	Monitored Device D2	Comments
9. Device D1 goes off hook and is connected on the call.	Originated • originatedConnection D1C3 • callingDevice D1 • calledDevice D2 • localConnectionInfo connected • cause CallBack • servicesPermitted ClearConn, SendUserInfo		
10.The failed connection of D2 is cleared.		Connection Cleared • droppedConnection D2C3 • releasingDevice D2 • localConnectionInfo null • cause normalClr • servicesPermitted none	
11.Device D2 is alerted.	Delivered • deliveredConnection D2C3 • alertingDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo connected • cause CallBack • servicesPermitted CallBack, ClearConn, SendUserInfo	Delivered • deliveredConnection D2C3 • alertingDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo alerting • cause CallBack • servicesPermitted ClearConn, Answer, Deflect, SendUserInfo	

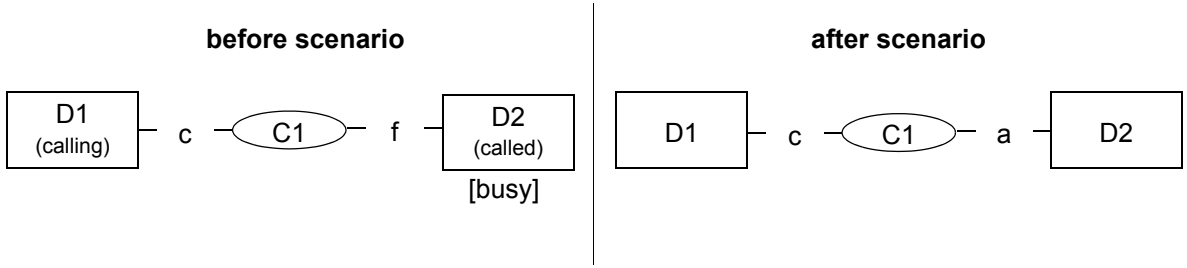
Table 5-55 Call Back Call Related (page 3 of 3)

Remark:

None

5.16.2 Manual Camp On Call

The calling party queues a call for a busy called device by pressing the camp-on key until that device becomes available.



See “Manually dialled call - called party is busy” on page 5-5 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1		Monitored Device D2		Comments
1. The busy connection is cleared immediately.	Connection Cleared		Connection Cleared		
	• droppedConnection	D2C1	• droppedConnection	D2C1	
	• releasingDevice	D2	• releasingDevice	D2	
	• localConnectionInfo	connected	• localConnectionInfo	null	
	• cause	normalClr	• cause	normalClr	
	• servicesPermitted	ClearConn	• servicesPermitted	none	
2. D1 presses the camp on key.	Queued		Queued		
	• queuedConnection	D2C1	• queuedConnection	D2C1	
	• queue	D2	• queue	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	queued	
	• cause	campOn	• cause	campOn	
	• servicesPermitted	CallBack, ClearConn, SendUserInfo	• servicesPermitted	SendUserInfo	
3. Some time later device D2 clears from its active call.			Connection Cleared		
			• droppedConnection	D2C2	
			• releasingDevice	D2	
			• localConnectionInfo	null	
			• cause	normalClr	
			• servicesPermitted	none	

Table 5-56 Manual Camp On Call (page 1 of 2)

Activity	Monitored Device D1		Monitored Device D2		Comments
4. Since device D2 is available the call alerts D2.	Delivered • deliveredConnection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted	 D2C1 D2 D1 D2 NS connected recall CallBack, ClearConn, SendUserInfo	Delivered • deliveredConnection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted	 D2C1 D2 D1 D2 NS alerting recall ClearConn, Answer, Deflect, SendUserInfo	

Table 5-56 Manual Camp On Call (page 2 of 2)

Remark:
None

5.17 Distribution Call Scenarios

5.17.1 Automatic Call Distribution Scenarios

5.17.1.1 Automatic Call Distribution Overview

5.17.1.1.1 ACD Call Processing

ACD Call Processing is based on a call analysis and routing scheme that processes calls according to customer requirements. The routing scheme is configured by defining ACD numbers, route control groups (RCGs), and ACD routing tables (ARTs) to process the call.

The sequence of call processing begins with ***"host-based" source- and destination-based routing,*** continues with ***calendar routing***, and is finally processed by the sequence of commands in the routing tables (***route processing***). Further routing may be necessary at the end of work shifts with ***end of shift routing***. For an illustration of this call process, see Figure 0-1 on page 5-112.

- "Host-based" Routing

  When a call arrives at the OpenScape 4000 destined for a call center, an external application may be used to route the call using the caller's identity, location, or the reason for the call. The application can take advantage of previous calls and place this call to an agent who has handled this customer previously. The application may route a caller to a different call center based on call volumes. The application could reside on a host or external server.

- Source- and Destination-based Routing

  Without special handling, when a call is placed to an ACD number, routing analysis begins when the ACD software determines the source and destination of the call. The source of the call is the number of the calling party received by the OpenScape 4000. It may be a 10-digit telephone number, the main number of the private branch exchange (PBX), or a private network number. Automatic Number Identification (ANI) is an example of a trunk service that provides a source number.

  The destination is the called party. It may be a translated Dialled Number Identification Service (DNIS), Direct Inward Dialling (DID), or in some cases, something other than the original number dialled by the calling party, converted by public or private network number translation. ACD uses this source and destination data to associate a route control group (RCG) with the call. The RCG defines routing to an ACD routing table, based on the time of day and the day of the week the call is processed.

- Calendar Routing

Calendar routing uses the day of the week and the time of day to reference the RCG and to select the ACD routing table (ART) for processing the call. Calendar Routing also can be used to setup holidays in advance.

- Route Processing

  The ACD routing table (ART) is used for final processing of the call. ACD routing tables contain a series of steps for processing the call. Each step in an ART is performed sequentially, except when a conditional step or a “GOTO” step directs the call to a specific step number. If this occurs, processing continues sequentially for the step defined in the GOTO statement. ARTs can be configured with two types of steps, fixed and conditional. Fixed steps route the call in a specific manner (for example, route to server). Conditional steps have dependencies.

  Each shift designates an ACD routing table to be used during that shift. As soon as the system assigns a call to an ART, it becomes an ACD call. A call made directly to the extension number of a logged on agent is not an ACD call.

  The computer application can activate or deactivate special handling of ACD calls using routing services. When a call comes into the OpenScape 4000, the computer application is notified, if special handling was set up. The computer application may route the call to a different destination based upon several factors: ANI number supplied, translated DNIS number; or the combination of ANI and DNIS numbers, if both are supplied. In these instances, ANI and DNIS get special routing instructions before normal processing.

- ACD Queuing

  To ensure that incoming calls are handled as efficiently as possible, a single call can queue for as many as 16 different ACD groups. CA 4000 can provide an application with events that monitor queuing activity.

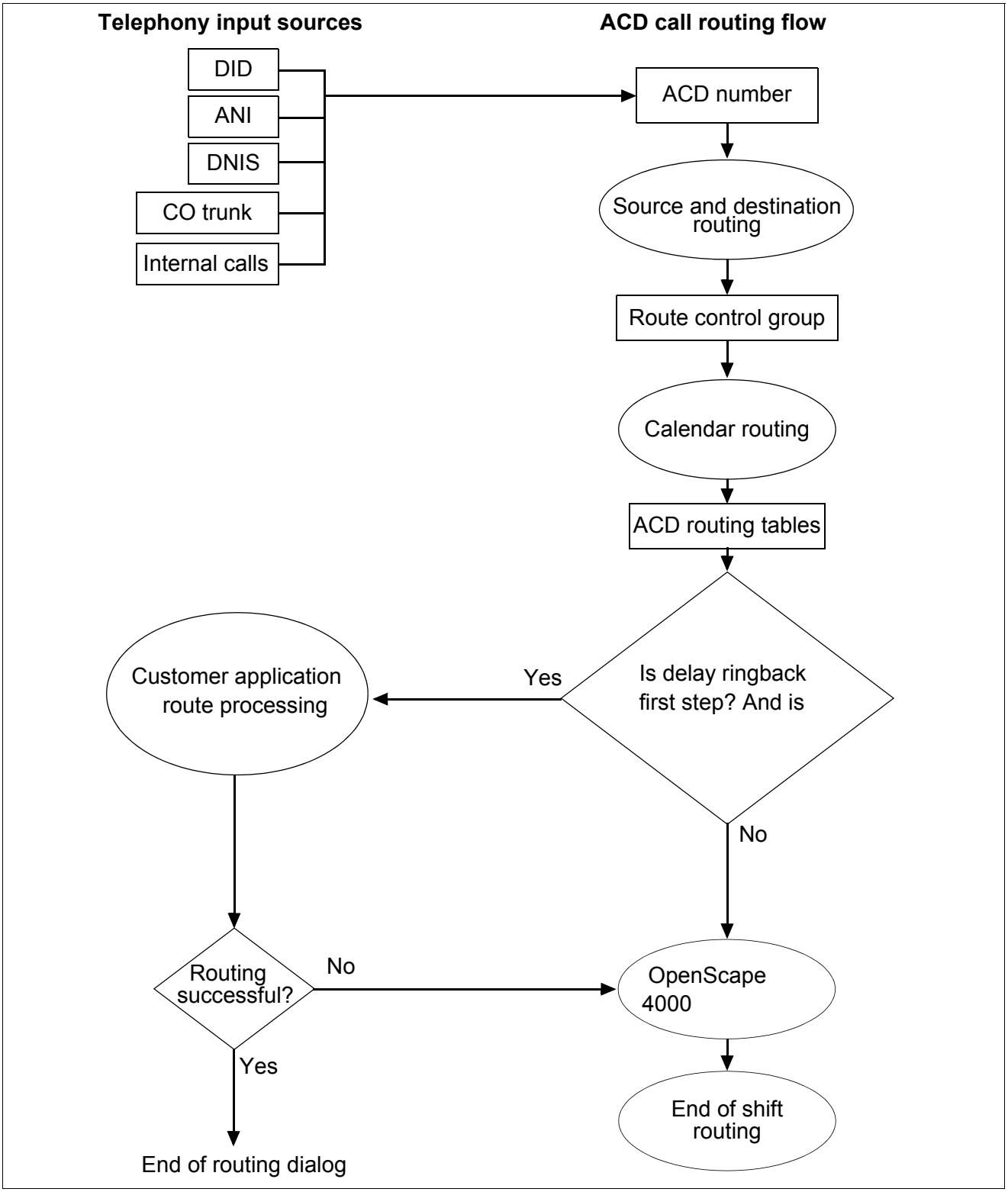


Figure 0-1 ACD Routing Flow Diagram

5.17.1.1.2 ACD Terminology

The following are some important ACD terms:

- ACD Calls

  An ACD call is an incoming call that reaches an ACD number. If an incoming call arrives on a trunk group dedicated to an ACD number, it immediately becomes an ACD call and begins to be routed to the system. However, if an incoming call arrives on a trunk that is not dedicated to an ACD number, the system does not route the call until it does reach an ACD number—for example, if the call is internally transferred.

- ACD Groups

  An ACD group is a group of agent extensions that receives calls.

  When a group member (agent) is busy on a call, the system routes the calls to an available agent within the group. If all agents are busy, the system can also transfer incoming calls from one ACD group to another.

  An ACD group can be a single extension, such as customer service. Larger departments can also be divided into many smaller ACD groups.

- ACD Number

  A diallable number that initiates processing of the call as an ACD call. The ACD number maps the call to a Route Control Group (RCG) depending on the source of the call (ANI), the destination of the call (DNIS), the day of the week, and the time of day.

- ACD RCG (Route Control Group)

  An Automatic Call Distribution (ACD) Route Control Group (RCG), configured in the OpenScape 4000 software, is the entry point used by ACD software for routing calls.

  An ACD number directs a call to an RCG. Each RCG is identified by a unique, non-diallable number that is defined by configuration. If the specified routing is destination, call processing uses the specified RCG for the destination ACD number, if the routing option is set to source, the system uses the source ACD number and its associated RCG for routing the call.

  The same RCG can be assigned to multiple ACD numbers.

- ACD Routing Table (ART)

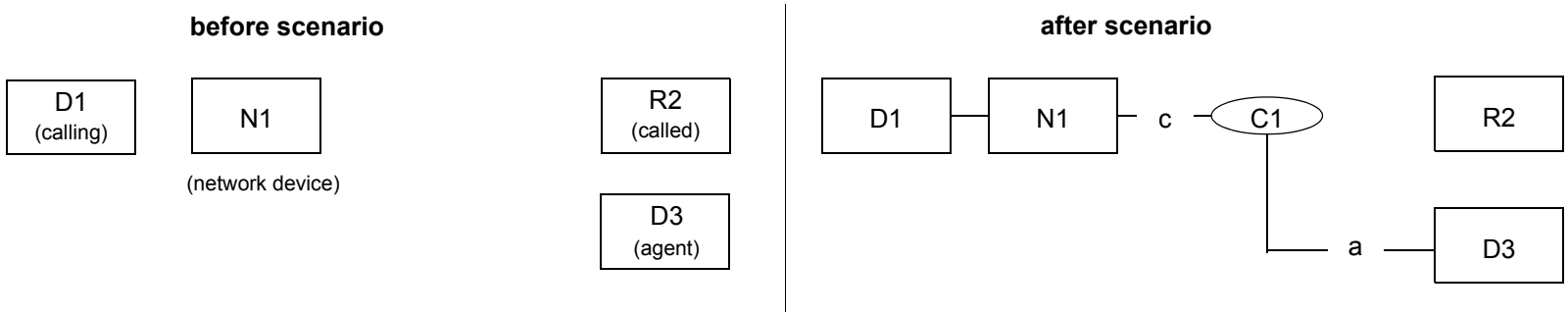
  ARTs are tables that permit the configuration of the call routing. An ACD routing table is a set of instructions that an ACD call follows until an agent is available to answer the call.

  For example, the caller may first hear a recorded message stating that all agents are still busy, followed by music for a certain number of seconds. If an agent is still not available, the call may be routed to another group of agents, or eventually to an off-site number.

Many routing tables can be configured for each ACD group. This permits customized routing to meet each group's requirements.

5.17.1.2 External ACD Call completed to agent

The external caller D1 (NID = N1) calls the ACD-pilot-number (R2 intDnis). No agent is available, the call is queued. As soon as an agent becomes available, the call is routed to ACD-group G. The call is routed to an agent.



Activity	Monitored Device N1 (Trunk)	Monitored Device R2 (RCG)	Monitored Device D3 (agent)	Comments
1. An incoming trunk is seized.	**Service Initiated** • initiatedConnection N1C1 • initiatingDevice N1 • localConnectionInfo initiated • cause normal • servicesPermitted ClearConn		None	

Table 5-57 External Call Incoming to ACD Agent – Call Completed to Agent (page 1 of 4)

Activity	Monitored Device N1 (Trunk)	Monitored Device R2 (RCG)	Monitored Device D3 (agent)	Comments
2. The call is routed to an ACD routing table .	**Originated** • originatedConnection N1C1 • callingDevice D1 • calledDevice R2 intDnis • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn			**(1) AssCledDev:** It is only provided for external calls to an RCG where the trunk is **not** monitored. It then contains the ACD-DNIS (e.g. R2 intDnis). In this scenario it is not provided because the trunk is monitored. **Please note:** this remark applies to all subsequent events for the RCG and the agent
	Delivered • connection R2C1 • alertingDevice R2 • callingDevice D1 • calledDevice R2 intDnis • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • • lastRedirectionDev NS • localConnectionInfo connected • cause enterDist • servicesPermitted ClearConn SendUI	**Delivered** • connection R2C1 • alertingDevice R2 • callingDevice D1 • calledDevice R2 intDnis • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • **AssCalledDevice NP (1)** • lastRedirectionDev NS • localConnectionInfo alerting • cause enterDist • servicesPermitted ClearConn Deflect SendUI		
3. The call is routed to ACD G group.	None	None	None	

Table 5-57 External Call Incoming to ACD Agent – Call Completed to Agent (page 2 of 4)

Activity	Monitored Device N1 (Trunk)	Monitored Device R2 (RCG)	Monitored Device D3 (agent)	Comments
4. The call is queued.	**Queued** • queuedConnection R2C1 • queue G • callingDevice D1 • calledDevice R2 intDnis • lastRedirectionDev NS • AssCallingDevice N1 • NWCallingDevice D1 • localConnectionInfo connected • cause NoAgents • servicesPermitted ClearConn SendUI	**Queued** • queuedConnection R2C1 • queue G • callingDevice D1 • calledDevice R2 intDnis • lastRedirectionDev NS • AssCallingDevice N1 • NWCallingDevice D1 • localConnectionInfo queued • cause NoAgents • servicesPermitted ClearConn Deflect SendUI	None	G is the ACD-group.
5. An agent becomes free; the call is diverted from the RCG.		**Diverted** • connection R2C1 • divertingDevice R2 • newDestination D3 • callingDevice D1 • calledDevice R2 intDnis • lastRedirectionDev NS • AssCallingDevice N1 • NWCallingDevice D1 • localConnectionInfo null • cause Distributed • servicesPermitted none • servicesPermitted none		

Table 5-57 External Call Incoming to ACD Agent – Call Completed to Agent (page 3 of 4)

Activity	Monitored Device N1 (Trunk)	Monitored Device R2 (RCG)	Monitored Device D3 (agent)	Comments
6. The call is delivered to the agent.	**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice R2 intDnis • lastRedirectionDev **NS** • OrigNIDConn N1C1 • AssCallingDevice N1 • NWCallingDevice D1 • localConnectionInfo connected • cause Distributed • servicesPermitted ClearConn SendUI		**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice R2 intDnis • lastRedirectionDev **NS** • OrigNIDConn N1C1 • AssCallingDevice N1 • NWCallingDevice D1 • localConnectionInfo alerting • cause Distributed • servicesPermitted Answer ClearConn Deflect SendUI	**Please note:** Even though the call was diverted from the RCG, the LastRedirectionDevice is not reported in the DeliveredEvents.

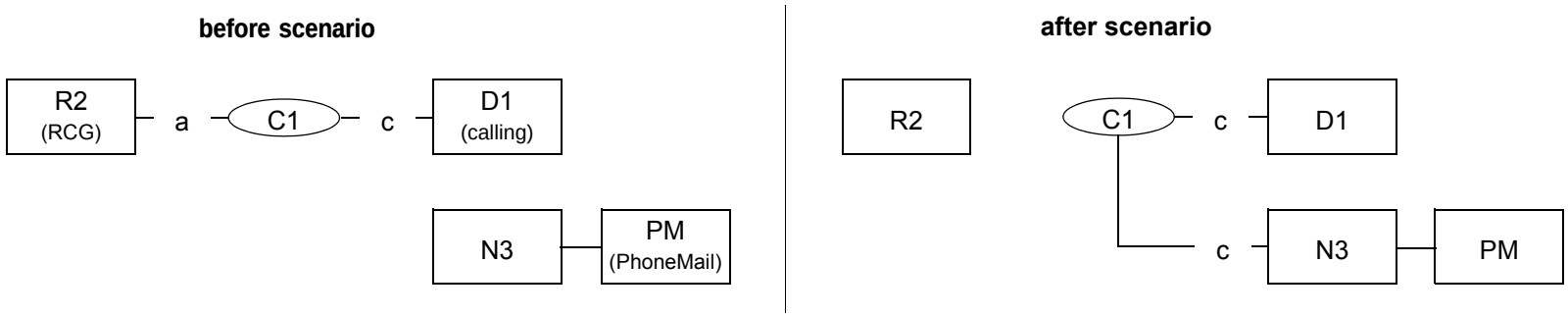
Table 5-57 External Call Incoming to ACD Agent – Call Completed to Agent (page 4 of 4)

Remark:

None

5.17.1.3 Internal ACD call completed to Phone Mail agent

This scenario describes a call flow when an RCG routes a call to a free Phonemail Agent.



Activity	Monitored Device D1	Monitored Device R2 (RCG)	Monitored Device N3	Comments
1. RCG R2 routes the call to the PM agent.		**Diverted** • connection R2C1 • divertingDevice R2 • newDestination PM • callingDevice D1 • calledDevice R2 pilot number • lastRedirectionDevice NS • localConnectionInfo null • cause distributed • servicesPermitted none		
2. The call leaves the CSTA subdomain.	**Network Reached** • outboundConnection N3C1 • networkInterfaceUsed N3 • callingDevice D1 • calledDevice R2 pilot number • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTone, SendUserInfo		**Network Reached** • outboundConnection N3C1 • networkInterfaceUsed N3 • callingDevice D1 • calledDevice R2 pilot number • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, SendUserInfo	
3. The Phone Mail answers the call immediatelly.	**Established** • establishedConnection N3C1 • answeringDevice PM • callingDevice D1 • calledDevice R2 pilot number • lastRedirectionDevice NS • localConnectionInfo connected • cause distributed • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTone, SendUserInfo • assocCalledDevice N3		**Established** • establishedConnection N3C1 • answeringDevice PM • callingDevice D1 • calledDevice R2 pilot number • lastRedirectionDevice NS • localConnectionInfo connected • cause distributed • servicesPermitted ClearConn, SendUserInfo • assocCalledDevice N3	

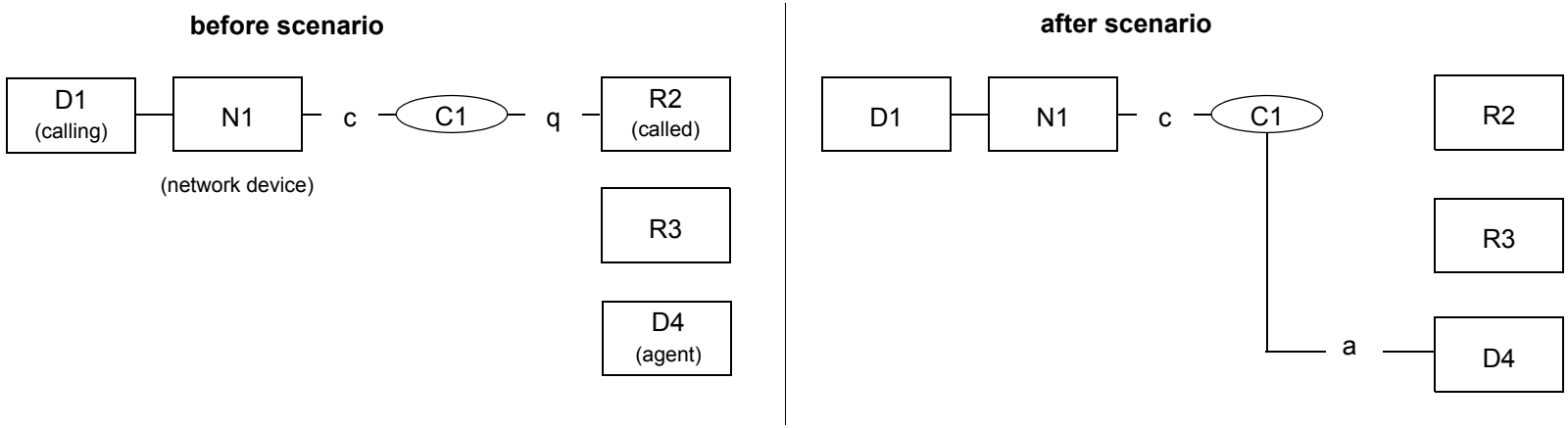
Table 5-58 Route to free Phone Mail Agent

Remark:

None

5.17.1.4 External call overflowed to another RCG

The external caller D1 (NID = N1) calls the ACD-pilot-number (R2 intDnis). No agent is available, the call is queued. After a time, the call is routed to another RCG (R3). An agent is available, the call is routed to the agent.



Activity	Monitored Device N1	Monitored Device R2	Monitored Device R3	Monitored Device D4 (agent)	Comments
1. The call was previously queued at RCG2. After a timer elapses, RCG2 diverts the call to RCG3	None	**Diverted** • connection R2C1 • divertingDevice R2 • newDestination R3 • callingDevice D1 • calledDevice R2 intDnis • lastRedirectionDev NS • AssCallingDevice N1 • NWCallingDevice D1 • localConnectionInfo null • cause normal • servicesPermitted none			

Table 5-59 External overflow to another RCG (page 1 of 3)

Activity	Monitored Device N1	Monitored Device R2	Monitored Device R3	Monitored Device D4 (agent)	Comments
2. The call rings at RCG3	**Delivered** • connection R3C1 • alertingDevice R3 • callingDevice D1 • calledDevice R2 intDnis • lastRedirectionDev NS • OrigNIDConn N1C1 • AssCallingDevice N1 • NWCallingDevice D1 • localConnectionInfo connected • cause enterDist • servicesPermitted ClearConn SendUI		**Delivered** • connection R3C1 • alertingDevice R3 • callingDevice D1 • calledDevice R2 intDnis • lastRedirectionDev NS • OrigNIDConn N1C1 • AssCallingDevice N1 • NWCallingDevice D1 • localConnectionInfo alerting • cause enterDist • servicesPermitted ClearConn Deflect SendUI		
3. The call is routed to an ACD Group	None	None	None	None	
4. An agent is available - the call is diverted from the RCG to the agent's phone			**Diverted** • connection R3C1 • divertingDevice R3 • newDestination D4 • callingDevice D1 • calledDevice R2 intDnis • lastRedirectionDev NS • AssCallingDevice N1 • NWCallingDevice D1 • localConnectionInfo null • cause Distributed • servicesPermitted none		

Table 5-59 External overflow to another RCG (page 2 of 3)

Activity	Monitored Device N1		Monitored Device R2	Monitored Device R3	Monitored Device D4 (agent)		Comments
5. The call starts ringing at the agent's phone	**Delivered**				**Delivered**		
	• connection	D4C1			• connection	D4C1	
	• alertingDevice	D4			• alertingDevice	D4	
	• callingDevice	D1			• callingDevice	D1	
	• calledDevice	R2 intDnis			• calledDevice	R2 intDnis	
	• lastRedirectionDev	NS			• lastRedirectionDev	NS	
	• OrigNIDConn	N1C1			• OrigNIDConn	N1C1	
	• assCallingDevice	N1			• assCallingDevice	N1	
	• NWCallingDevice	D1			• NWCallingDevice	D1	
	• localConnectionInfo	connected			• localConnectionInfo	alerting	
	• cause	Distributed			• cause	Distributed	
	• servicesPermitted	ClearConn SendUI			• servicesPermitted	Answer ClearConn Deflect SendUI	

Table 5-59 External overflow to another RCG (page 3 of 3)

Remark:

None

5.17.2 Make Predictive Call

The Make Predictive Call Service originates a call between two devices by first creating a connection to the called device. The service returns a positive acknowledgment that provides the connection at the called device.

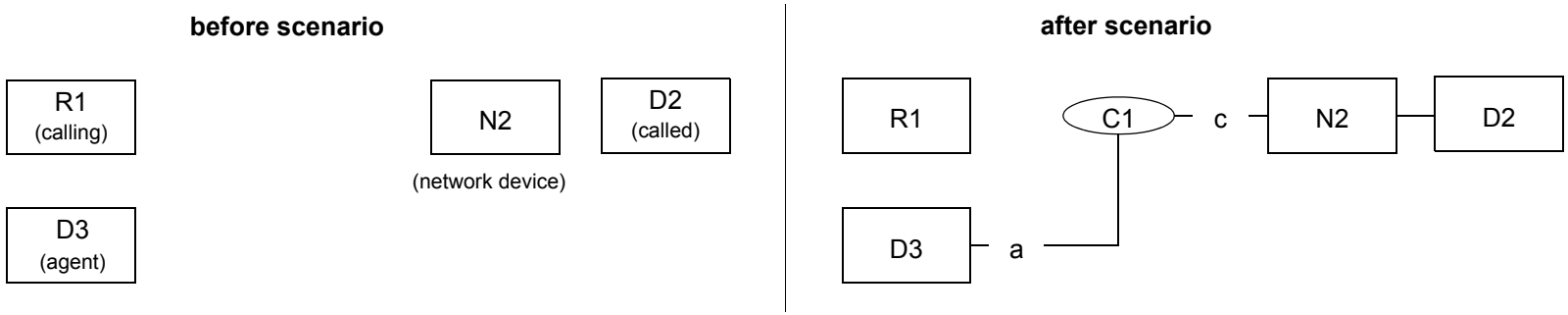
On OpenScape 4000, the calling device is always an RCG. The called device may be any station or trunk. After the called device has answered the call, the RCG proceeds with its ART-Table. Usually, the call is either distributed to an agent or diverted to a station.

Connections created by Make Predictive Call are cleared after:

- the called party fails to answer within a certain amount of time
- the switch has detected that the called device is unable to answer (is busy, for example)

5.17.2.1 Make Predictive Call - to external free device

Make Predictive Call from RCG to external party outside the CSTA subdomain. The external party answers the call, the RCG routes the call to an agent.



Activity	Monitored Device R1 (RCG)	Monitored Device N2 (trunk)	Monitored Device D3 (agent)	Comments
1. A Make Predictive Call to a valid device is invoked on behalf of a RCG	**Make Predictive Call - Service Request** • callingDevice R1 • calledDirectoryNumber D2 **Make Predictive Call - Positive Response.** • initiatedCall N2C1			
2. RCG device is initiated	**Service Initiated** • initiatedConnection R1C1 • initiatingDevice R1 • localConnectionInfo initiated • cause makePredCall • servicesPermitted ClearConn			

Table 5-60 Make Predictive Call - to external free device (page 1 of 4)

Activity	Monitored Device R1 (RCG)		Monitored Device N2 (trunk)		Monitored Device D3 (agent)	Comments
3. The call leaves the CSTA subdomain	**Network Reached**		**Network Reached**			
	outboundConn	N2C1	outboundConn	N2C1		
	NWInterfaceUsed	N2	NWInterfaceUsed	N2		
	callingDevice	R1	callingDevice	R1		
	calledDevice	D2	calledDevice	D2		
	lastRedirectionDev	NS	lastRedirectionDev	NS		
	NW-Capability	ISDN Public	NW-Capability	ISDN Public		
	localConnectionInfo	initiated	localConnectionInfo	connected		
	cause	normal	cause	normal		
	servicesPermitted	ClearConn	servicesPermitted	ClearConn, Deflect SendUI		
4. D2 is alerted	**Delivered**		**Delivered**			
	• connection	N2C1	• connection	N2C1		
	• alertingDevice	D2	• alertingDevice	D2		
	• callingDevice	R1	• callingDevice	R1		
	• calledDevice	D2	• calledDevice	D2		
	• AssCalledDevice	N2	• AssCalledDevice	N2		
	• lastRedirectionDev	NS	• lastRedirectionDev	NS		
	• localConnectionInfo	initiated	• localConnectionInfo	connected		
	• cause	networkSignal	• cause	networkSignal		
	• servicesPermitted	ClearConn	• servicesPermitted	ClearConn, Deflect SendUI		
5. D2 answers the call	**Established**		**Established**			
	• establishedConn	N2C1	• establishedConn	N2C1		
	• answeringDevice	D2	• answeringDevice	D2		
	• callingDevice	R1	• callingDevice	R1		
	• calledDevice	D2	• calledDevice	D2		
	• AssCalledDevice	N2	• AssCalledDevice	N2		
	• lastRedirectionDev	NS	• lastRedirectionDev	NS		
	• localConnectionInfo	initiated	• localConnectionInfo	connected		
	• cause	networkSignal	• cause	networkSignal		
	• servicesPermitted	ClearConn, SendUI	• servicesPermitted	ClearConn, SendUI		

Table 5-60 Make Predictive Call - to external free device (page 2 of 4)

Activity	Monitored Device R1 (RCG)	Monitored Device N2 (trunk)	Monitored Device D3 (agent)	Comments
6. Call comes back into the switching domain and is delivered to the RCG	**Delivered** • connection R1C1 • alertingDevice R1 • callingDevice R1 • calledDevice D2 • AssCalledDevice N2 • lastRedirectionDev NS • localConnectionInfo alerting • cause enteringDist • servicesPermitted ClearConn, SendUI	**Delivered** • connection R1C1 • alertingDevice R1 • callingDevice R1 • calledDevice D2 • AssCalledDevice N2 • lastRedirectionDev NS • localConnectionInfo connected • cause enteringDist • servicesPermitted ClearConn SendUI		
7. An Available agent at Device D3 is chosen and the call is diverted to that device.	**Diverted** • connection R1C1 • divertingDevice R1 • newDestination D3 • **AssCalledDevice N2** • lastRedirectionDev NS • localConnectionInfo null • cause distributed • servicesPermitted none			**Please note:** the calledDevice is optional and not provided. However, the AssCalledDevice is mandatory for outgoing external calls and therefore provided.
8. D3 is alerted		**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D3 • calledDevice D2 • AssCalledDevice N2 • lastRedirectionDev NS • localConnectionInfo connected • cause distributed • servicesPermitted ClearConn SendUI	**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D3 • calledDevice D2 • AssCalledDevice N2 • lastRedirectionDev NS • localConnectionInfo alerting • cause distributed • servicesPermitted Answer ClearConn Deflect SendUI	

Table 5-60 Make Predictive Call - to external free device (page 3 of 4)

Activity	Monitored Device R1 (RCG)	Monitored Device N2 (trunk)		Monitored Device D3 (agent)		Comments
9. D3 answers the call		**Established**		**Established**		
		• establishedConn	D3C1	• establishedConn	D3C1	
		• answeringDevice	D3	• answeringDevice	D3	
		• callingDevice	D3	• callingDevice	D3	
		• calledDevice	D2	• calledDevice	D2	
		• AssCalledDevice	N2	• AssCalledDevice	N2	
		• lastRedirectionDev	NS	• lastRedirectionDev	NS	
		• localConnectionInfo	connected	• localConnectionInfo	connected	
		• cause	NWSignal	• cause	NWSignal	
		• servicesPermitted	ClearConn, SendUI	• servicesPermitted	ClearConn, Consultation, Hold, SST, GenDG, GenTelTone,SendUI	

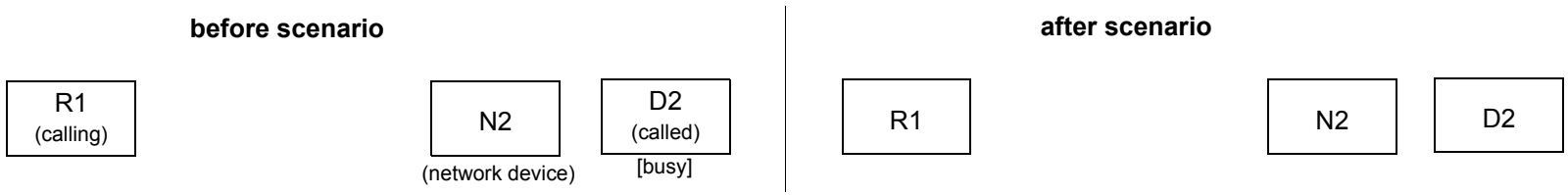
Table 5-60 Make Predictive Call - to external free device (page 4 of 4)

Remark:

None

5.17.2.2 Make Predictive Call - to external busy device

Make Predictive Call to a busy party outside the CSTA subdomain. The call cannot be completed.



Activity	Monitored Device R1		Monitored Device N2 (trunk)		Comments
1. A Make Predictive Call to a valid device is invoked on behalf of a RCG	**Make Predictive Call - Service Request**				
	• callingDevice	R1			
	• calledDirectoryNumber	D2			
	Make Predictive Call - Positive Response				
	• initiatedCall	N2C1			
2. RCG device is initiated	**Service Initiated**				
	• initiatedConnection	R1C1			
	• initiatingDevice	R1			
	• localConnectionInfo	initiated			
	• cause	makePredCall			
	• servicesPermitted	none			
3. The call leaves the CSTA subdomain	**Network Reached**		**Network Reached**		
	• outboundConn	N2C1	• outboundConn	N2C1	
	• NWInterfaceUsed	N2	• NWInterfaceUsed	N2	
	• callingDevice	R1	• callingDevice	R1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDev	NS	• lastRedirectionDev	NS	
	• NW-Capability	ISDN Private	• NW-Capability	ISDN Private	
	• localConnectionInfo	initiated	• localConnectionInfo	connected	
	• cause	normal	• cause	normal	
	• servicesPermitted	none	• servicesPermitted	none	
4. Device D2 is busy the call cannot be completed.	**Failed**		**Failed**		
	• failedConnection	N2C1	• failedConnection	N2C1	
	• failingDevice	D2	• failingDevice	D2	
	• callingDevice	R1	• callingDevice	R1	
	• calledDevice	D2	• calledDevice	D2	
	• AssCalledDevice	N2	• AssCalledDevice	N2	
	• lastRedirectionDev	NS	• lastRedirectionDev	NS	
	• localConnectionInfo	initiated	• localConnectionInfo	fail	
	• cause	busy	• cause	busy	
	• servicesPermitted	none	• servicesPermitted	none	

Table 5-61 Make Predictive Call - to external busy device (page 1 of 2)

Activity	Monitored Device R1		Monitored Device N2 (trunk)		Comments
5. Connection is cleared for device D2	**Connection Cleared** • droppedConnection • releasingDevice • localConnectionInfo • cause • servicesPermitted	 N2C1 N2 initiated normalClr none	**Connection Cleared** • droppedConnection • releasingDevice • localConnectionInfo • cause • servicesPermitted	 N2C1 N2 null normalClr none	
6. Connection is cleared for device D1 (RCG)	**Connection Cleared** • droppedConnection • releasingDevice • localConnectionInfo • cause • servicesPermitted	 R1C1 R1 null normalClr none			Connection clears as a result of Make Predictive Call condition

Table 5-61 Make Predictive Call - to external busy device (page 2 of 2)

Remark:

None

5.17.3 Route Services

The OpenScape 4000 is capable of allowing an external computer application to influence incoming calls. The application may divert incoming calls to a different agent, ACD group, or to another point within the telephone network. If the diverted destination is busy, the OpenScape 4000 allows the computer application further opportunities to reroute the call.

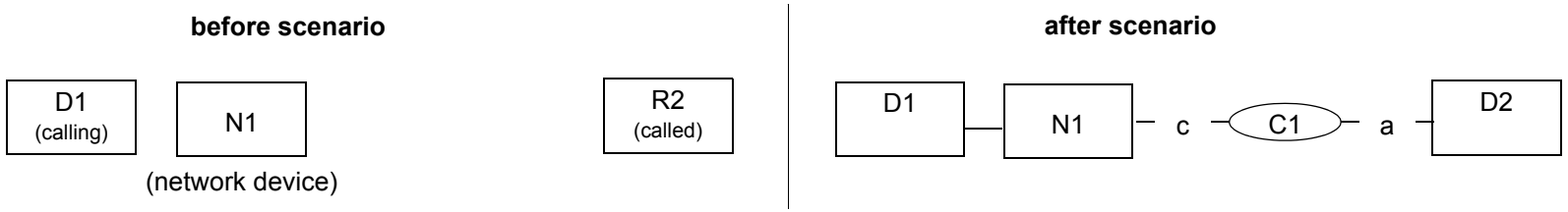
For an application to influence a call, two requirements must be met. First, the RCG to which the call was originally destined must be registered by the application or the gateway. The second requirement is that the ACD routing table (ART) within the RCG must have a delay ringback as its first programmed sequence. The delay ringback step has a timer associated with it that determines the length of time that the application may influence the call. If these two requirements are met, the application can influence the call.

The delay ringback timer allows the computer application time to influence the routing of the incoming call before the OpenScape 4000 sends the caller ringback tone. Each time a call comes into a registered RCG, the OpenScape 4000 sends a Route Request to the gateway, thereby giving the computer application an opportunity to influence the call. The Route Request initiates a routing dialog with the application. The gateway has 5 seconds to respond. The response time is measured by a routing timer within the OpenScape 4000. If the routing timer expires, the OpenScape 4000 sends the gateway a Route End signal to terminate the routing dialog. If the application responds to the OpenScape 4000 Route Request with a Route Select request within the required time, the OpenScape 4000 attempts to divert the call to the new destination specified by the computer application. If the OpenScape 4000 is successful in diverting the call, it sends a Route End signal to the application, terminating the routing dialog. If the OpenScape 4000 is not successful in diverting the call to the destination specified by the application, such as when the specified destination is busy, the OpenScape 4000 sends a Re-Route Request to the gateway. The route timer is reset to 5 seconds, and the application can initiate a new Route Select with a new destination. Sometimes, the application may not influence the call when it receives the Route Request from the OpenScape 4000. Instead of responding with a Route Select signal to the OpenScape 4000, it sends a Route End signal terminating the routing dialog. In this instance, the OpenScape 4000 continues processing the incoming call by using the next step on the RCG's ART table. In some cases, the application may reject the incoming call. If the ADR service is purchased and programmed, the network will redirect the call to another destination at some other location on the telephone network.

The computer application may route the incoming call to another RCG that is different from the RCG that the call was originally destined for. If the second RCG is registered and has delay ringback set in its ART table, call processing will bypass the delay ringback step and proceed to the next step in the ART table.

5.17.3.1 Route Request Scenario

This scenario describes an event flow of an external incoming call to an RCG that is redirected by the computer application to a different destination.



Activity	Monitored Device N1		Monitored Device R2 (RCG)	Monitored Device D2	Comments
1. N1 seized.	**Service Initiated**				
	• initiatedConnection	N1C1			
	• initiatingDevice	N1			
	• localConnectionInfo	initiated			
	• cause	normal			
	• servicesPermitted	ClearConn			
2. N1 completes dialling the R2 RCG.	**Originated**				
	• originatedConnection	N1C1			
	• callingDevice	D1			
	• calledDevice	R2 intDNIS			
	• localConnectionInfo	connected			
	• cause	normal			
	• servicesPermitted	ClearConn			
	• networkCallingDevice	D1			
	• assocCallingDevice	N1			
	• assocCalledDevice	R2 intDNIS			

Table 5-62 Route Request scenario (page 1 of 3)

Activity	Monitored Device N1		Monitored Device R2 (RCG)		Monitored Device D2	Comments
3. The call arrives at the RCG.	**Delivered**		**Delivered**			
	• deliveredConnection	R2C1	• deliveredConnection	R2C1		
	• alertingDevice	R2	• alertingDevice	R2		
	• callingDevice	D1	• callingDevice	D1		
	• calledDevice	R2 intDNIS	• calledDevice	R2 intDNIS		
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS		
	• originalNID	N1C1	• originalNID	N1C1		
	• localConnectionInfo	connected	• localConnectionInfo	alerting		
	• cause	enterDistribution	• cause	enterDistribution		
	• servicesPermitted	ClearConn, SendUserInfo	• servicesPermitted	ClearConn, Deflect		
	• networkCallingDevice	D1	• networkCallingDevice	D1		
	• assocCallingDevice	N1	• assocCallingDevice	N1		
	• assocCalledDevice	R2 intDNIS	• assocCalledDevice	R2 intDNIS		
4. Route Request is sent to the application to let the computing function route the call.	**Route Request**					
	• crossRefID	1				
	• referenceID	1				
	• calledDevice	R2 intDNIS				
	• callingDevice	D1				
	• routingDevice	R2				
	• routedCall	R2C1				
	• assocCallingDevice	N1				
	• assocCalledDevice	R2 intDNIS				
5. The application sends the route destination.	**Route Select Request**					
	• crossRefID	1				
	• routeRegisterRequestID	1				
	• routeSelected	D2				

Table 5-62 Route Request scenario (page 2 of 3)

Activity	Monitored Device N1	Monitored Device R2 (RCG)	Monitored Device D2	Comments
6. The routing is successful.		**Diverted** • divertedConnection R2C1 • divertingDevice R2 • newDestinationDevice D2 • callingDevice D1 • calledDevice R2 intDNIS • lastRedirectionDevice NS • localConnectionInfo null • cause normal • servicesPermitted SendUserInfo • networkCallingDevice D1 • assocCallingDevice N1 • assocCalledDevice R2 intDNIS		The switching function sends the Diverted event only to the divertingDevice.
7. D2 alerts.	**Delivered** • deliveredConnection D2C1 • alertingDevice D2 • callingDevice D1 • calledDevice R2 intDNIS • lastRedirectionDevice NS • originalNID N1C1 • localConnectionInfo connected • cause distributed • servicesPermitted CallBack, ClearConn, SendUserInfo • networkCallingDevice D1 • assocCallingDevice N1 • assocCalledDevice R2 intDNIS		**Delivered** • deliveredConnection D2C1 • alertingDevice D2 • callingDevice D1 • calledDevice R2 intDNIS • lastRedirectionDevice NS • originalNID N1C1 • localConnectionInfo alerting • cause distributed • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo • networkCallingDevice D1 • assocCallingDevice N1 • assocCalledDevice R2 intDNIS	The switching function provides lastRedirectionDevice NS instead of the proper value.
8. Route End Request is sent to the application to close the dialog.	**Route End Request** • crossRefID 1 • routeRegisterRequestID 1			

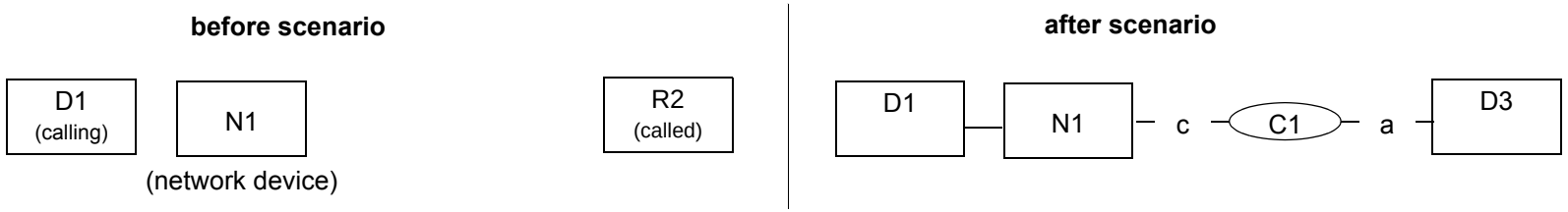
Table 5-62 Route Request scenario (page 3 of 3)

Remark:

None

5.17.3.2 Re-Route Request Scenario

This scenario describes an event flow of an external incoming call to an RCG that is redirected by the computer application to a busy destination and after that, it is rerouted to another destination. No events will be reported for the busy destination.



Activity	Monitored Device N1	Monitored Device R2 (RCG)	Monitored Device D3	Comments
Steps 1-3 are shown in "Route Request Scenario" on page 5-129.				
4. Route Request is sent to the application to let the computing function route the call.	**Route Request** • crossRefID 1 • referenceID 1 • calledDevice R2 intDNIS • callingDevice D1 • routingDevice R2 • routedCall R2C1 • assocCallingDevice N1 • assocCalledDevice R2 intDNIS			
5. The application sends the route destination.	**Route Select Request** • crossRefID 1 • routeRegisterRequestID 1 • routeSelected D2			

Table 5-63 Re-Route Request scenario (page 1 of 3)

Activity	Monitored Device N1	Monitored Device R2 (RCG)	Monitored Device D3	Comments
6. ReRoute Request is sent to the application to inform it that the routing destination was busy and to let the computing function choose a new destination.	**ReRoute Request** • crossRefID 1 • routeRegisterRequestID 1			
7. The application sends its other destination.	**Route Select Request** • crossRefID 1 • routeRegisterRequestID 1 • routeSelected D3			The new destination becomes D3.
8. The routing is successful.		**Diverted** • divertedConnection R2C1 • divertingDevice R2 • newDestinationDevice D3 • callingDevice D1 • calledDevice R2 intDNIS • lastRedirectionDevice NS • localConnectionInfo null • cause normal • servicesPermitted SendUserInfo • networkCallingDevice D1 • assocCallingDevice N1 • assocCalledDevice R2 intDNIS		The switching function sends the Diverted event only to the divertingDevice.

Table 5-63 Re-Route Request scenario (page 2 of 3)

Activity	Monitored Device N1		Monitored Device R2 (RCG)	Monitored Device D3		Comments
9. D2 alerts.	**Delivered**			**Delivered**		The switching function provides lastRedirectionDevice NS instead of the proper value.
	• deliveredConnection	D3C1		• deliveredConnection	D3C1	
	• alertingDevice	D3		• alertingDevice	D3	
	• callingDevice	D1		• callingDevice	D1	
	• calledDevice	R2 intDNIS		• calledDevice	R2 intDNIS	
	• lastRedirectionDevice	NS		• lastRedirectionDevice	NS	
	• originalNID	N1C1		• originalNID	N1C1	
	• localConnectionInfo	connected		• localConnectionInfo	alerting	
	• cause	distributed		• cause	distributed	
	• servicesPermitted	CallBack, ClearConn, SendUserInfo		• servicesPermitted	Answer, ClearConn, Deflect, SendUserInfo	
	• networkCallingDevice	D1		• networkCallingDevice	D1	
	• assocCallingDevice	N1		• assocCallingDevice	N1	
	• assocCalledDevice	R2 intDNIS		• assocCalledDevice	R2 intDNIS	
10.Route End Request is sent to the application to close the dialog.	**Route End Request**					
	• crossRefID	1				
	• routeRegisterRequestID	1				

Table 5-63 Re-Route Request scenario (page 3 of 3)

Remark:

None

5.17.3.3 Route End Request Scenario

This scenario describes an event flow of an external incoming call to an RCG for which the computer application is allowed to influence the destination routing, but the computer application declines to use its rerouting. The next step in the RCG's ACD routing table will be selected.

Activity	Monitored Device N1	Monitored Device N2	Comments
Steps 1-3 are shown in "Route Request Scenario" on page 5-129.			

Table 5-64 Route End Request scenario (page 1 of 2)

Activity	Monitored Device N1		Monitored Device N2	Comments
4. Route Request is sent to the application to let the computing function route the call.	**Route Request** • crossRefID • referenceID • calledDevice • callingDevice • routingDevice • routedCall • assocCallingDevice • assocCalledDevice	 1 1 R2 intDNIS D1 R2 R2C1 N1 R2 intDNIS		
5. The application sends Route End Request.	**Route End Request** • crossRefID • routeRegisterRequestID	 1 1		

Table 5-64 Route End Request scenario (page 2 of 2)

Remark:

None

5.17.3.4 Reject Call Scenario

The Reject Call request is sent by the computer application to the switching function during a routing dialog to indicate that the OpenScape 4000 should return busy to the network. If the customer has purchased the Alternate Destination Redirection (ADR) service from the network, the ADR service causes the incoming call to be routed to another (preconfigured) destination in the network.

Activity	Monitored Device N1	Monitored Device R2 (RCG)	Comments
Steps 1-3 are shown in "Route Request Scenario" on page 5-129.			

Table 5-65 Reject Call scenario (page 1 of 2)

Activity	Monitored Device N1		Monitored Device R2 (RCG)		Comments
4. Route Request is sent to the application to let the computing function route the call.	**Route Request**				
	• crossRefID	1			
	• referenceID	1			
	• calledDevice	R2 intDNIS			
	• callingDevice	D1			
	• routingDevice	R2			
	• routedCall	R2C1			
	• assocCallingDevice	N1			
	• assocCalledDevice	R2 intDNIS			
5. The application rejects the call.	**Reject Request**				
	• crossRefID	1			
	• routeRegisterRequestID	1			
6. Route End Request will be sent to the application to close the dialog.	**Route End Request**				
	• crossRefID	1			
	• routeRegisterRequestID	1			
7. RCG clears the far end will get busy tone.	**Connection Cleared**		**Connection Cleared**		
	• droppedConnection	R2C1	• droppedConnection	R2C1	
	• releasingDevice	R2	• releasingDevice	R2	
	• localConnectionInfo	connected	• localConnectionInfo	null	
	• cause	normalClr	• cause	normalClr	
	• servicesPermitted	ClearConn	• servicesPermitted	none	
8. D1 hungs up.	**Connection Cleared**				
	• droppedConnection	N1C1			
	• releasingDevice	N1			
	• localConnectionInfo	null			
	• cause	normalClr			
	• servicesPermitted	none			

Table 5-65 Reject Call scenario (page 2 of 2)

Remark:

None

5.17.4 Hunting Groups (HG)

5.17.4.1 General description HG

5.17.4.1.1 Introduction

Monitoring of Hunting Groups was provided in a non-standard way in previous versions. Please note that the call-flow has changed considerably!

CSTA III (ECMA 269) and the ECMA Call-Scenarios do not provide much information about how devices like HG should be modeled. Therefore it is necessary to describe the model CA 4000 uses.

5.17.4.1.2 Characteristics of HG

A Hunting Group is represented by a logical device, it does not have a physical appearance. Each HG is assigned at least one diallable number and a (unique) device-number. To monitor the HG, the device-number must be used (similar to RCGs, the number issued by HiPath 4000 must be masked). The mask for HG is 0x05000000. It is **not** possible to monitor the diallable number of a HG.

A HG has members and - usually - a queue where incoming calls are queued when no member is available. A monitor on the HG reports events for the HG-Device only. If an application wants to receive events for the members as well, it is necessary that it monitors all members individually. This is called the „Group-Exclusive-Model“.

Remark: an application may obtain information about the members of a HG by invoking the GetLogicalDeviceInformation-Request.

If only the member of a HG is monitored but not the HG itself, an application will be able to tell the difference between a HG-call and a direct call to the member by looking at the event-cause in the Delivered-Event: if the cause is „Multi Alert“ or „RemainsInQueue“, the call has been distributed by the HG.

If a party picks up a HG-call ringing at a HG-member, the application needs to interpret the CSTA-CalledDevice to be able to tell whether the call was originally for a HG.
Please note: this piece of information is not reliable, because the called device could be a device forwarded to the HG.

The main task of a HG is the distribution of calls to its members. In most cases, the HG remains involved in the call until

- the member answers the call or
- the caller has hung up

This means that both the HG and its member are ringing simultaneously. For handling these situations, CSTA III has introduced the new event-cause „MultipleAlerting“. Please refer to 5.17.4.1.6, “General Rules concerning Multi-Alert-Situations” for more information.

5.17.4.1.3 Deflect and HG

The following has to be considered when using the Deflect-Call-Request for calls where a HG is involved:

- For HG, Deflect-Call is only allowed when the call is queued
- Deflect-Call is **not** allowed in a Multiple-Alert-Situation (this includes **all** devices involved in the call).

5.17.4.1.4 Special features of HG

- **HG-member:** usually a normal station; the HG distributes incoming calls to its members. The type of distribution (linear, cyclic) is configured in the switch. When a HG-member is ringing with a HG-call, two devices are ringing simultaneously: the HG and the HG-member. This is reflected in the Delivered-Event by the event-cause Multi-Alert. This indicates that the HG is still in control of the call.
  Please note: a HG-member can be on another node. In that case the used trunk will be treated as HG-member.
- **Control of the call**: the HG remains in control of the call until the call has left the HG for one of the following reasons:
  - the HG-member has answered the call
  - another device has picked up a call ringing at a HG-member
  - the caller has hung up
  - the call is deflected to another destination (only permitted when the call is in the HG-queue)
- **Hunt Advance**: a HG member fails to answer the call in time; the HG releases the device not answering the call and distributes the call to its next member
- **HG-Queue:** if a HG has no member configured or if no member is available, the call is queued in the HG-Queue. This is the only situation when HiPath 4000 permits a Deflect from a HG.
- **Overflow-Destination**: if the HG-Queue is full, HiPath 4000 seizes the overflow-destination for the HG (if configured). In this special case, HG withdraws from the call as soon as the overflow-destination starts ringing.

5.17.4.1.5 Known Restrictions for HG

5.17.4.1.5.1 Event-cause RemainInQ vs. MultiAlert for Non-Group-device

If a call is distributed to a HG or a hunt-advance is performed, the status of the calling device in the HG remains unchanged: when the caller was queued, the position in the queue is kept. When the calling device was alerting at the HG, it stays alerting.

For the Delivered-Event there are two different event-causes that indicate whether the caller is queued or alerting at the Group-Device:

- Remains in Queue: the caller was queued before and keeps the position in the queue
- Multi Alert: the caller was alerting and is still alerting at the HG

Event-cause „RemainInQ“ can only be provided for the calling and called party (= group member) if the HG is monitored. Otherwise, event-cause will be Multi Alert in both cases. This means e.g. that the calling party will receive a Queued-Event when the call is queued at the group-device and afterwards a Delivered-Event with event-cause MultiAlert when the call is distributed to the first HG-member, but physically the call will remain in the queue (refer to section 5.17.4.3, “Internal call to HG, Hunt Advance” and section 5.17.4.4, “Call is queued at HG”).

5.17.4.1.5.2 ACD routes MakePredictiveCall-call into a HG

If a call generated by Make-Predictive-Call is routed to a HG by the RCG (instead of an agent), the CallingDevice cannot be provided any longer as soon as the call hits the Group-Device and is therefore reported as „NotKnown“.

5.17.4.1.5.3 Called-Device

There are situations where CA 4000 reports „NotKnown“ as Called-Device when a HG is involved. Some of these situations are:

- SST into a HG
- Deflect into a HG (when the HG is the first device to be monitored)

5.17.4.1.6 General Rules concerning Multi-Alert-Situations

5.17.4.1.6.1 What is a Multi-Alert-Situation?

Multi-Alerting means that a call is ringing at more than one devices. For HG, this happens when the HG distributes a call to one of its members. Both the member and the Group-Device are ringing simultaneously (see section 5.17.4.3, “Internal call to HG, Hunt Advance”)

5.17.4.1.6.2 Rules for Multi-Alert-Situations concerning HG

Generally, CA 4000 models diversions by sending the Diverted-Event for the Diverting-Device only. In case of a successful diversion, the calling device and the new destination will receive a Delivered-Event with LastRedirectionDevice = Diverting Device.

Here are some rules CA 4000 uses for handling Diversions / Multi-Alert-situations. Please note: these rules are not described in the ECMA CSTA III standard:

1. If a call is ringing at a device B and CA 4000 sends another Delivered-Event with a new AlertingDevice C, an application must infer that a diversion from B to C has taken place - unless the event cause is MultiAlert. (see rule # 2)

   Examples:

- Deflect from B to C
- CallForward-NoAnswer from B to C

2. If the second Delivered-Event is sent with event-cause „MultiAlert“, this means that a new alerting device has been added to the call. In this case, no diversion has taken place. Now, more than one devices are ringing simultaneously with the same call.

   Example:

   - HG distributes the call to its member; both the HG and the member are ringing (see section 5.17.4.3, “Internal call to HG, Hunt Advance”)

3. If an alerting device leaves a Multi-Alert-call and reduces the Multiple-Alerting to a „normal“ alerting, it depends upon the situation whether Diverted or Conn-Cleared is sent for the device leaving the call.
   Diverted is only sent if the call has moved to a new destination that was not involved with the call before.

   Example:

   - Another party picks up a HG-call ringing at a HG-member (see section 5.17.4.7, “Pick from HG-member”)
     In all other situations, a Connection-Cleared-Event must be sent.

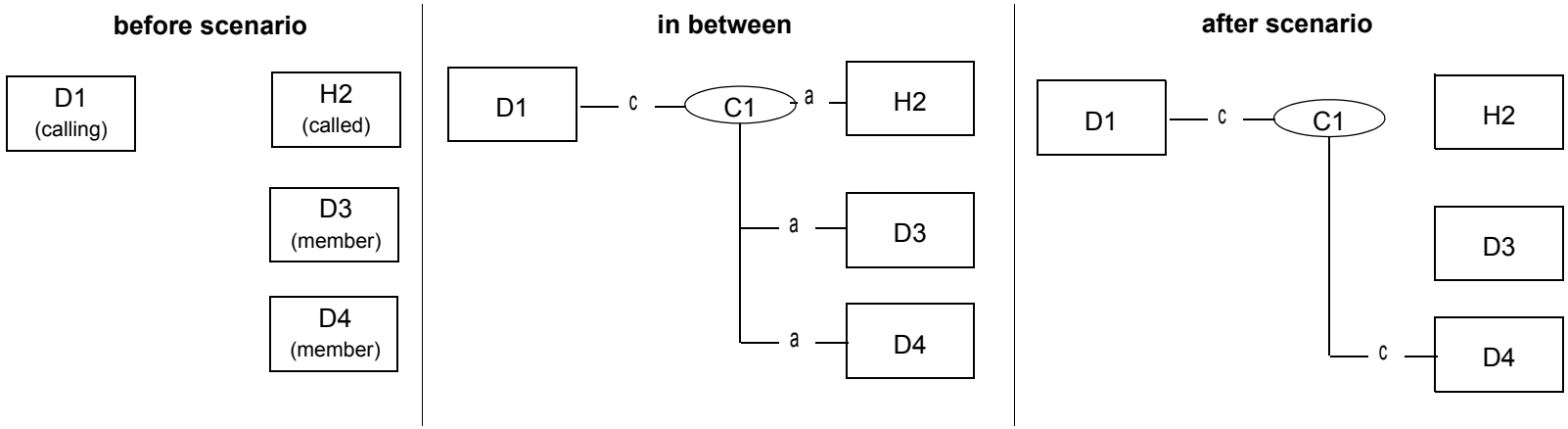
   Example:

   - The HG leaves the call because the HG-member has answered the call; a Conn-Cleared has to be sent for the HG (see section 5.17.4.3, “Internal call to HG, Hunt Advance”).

4. Event-cause „RemainsInQueue“: in special situations this event-cause is reported instead of „Multi-Alert“: whenever a call was placed in the queue of a HG, the call remains in the queue until the HG leaves the call. In these cases, the event-cause „Remains In Queue“ will be reported instead of „Multiple Alerting“ to show that the call keeps its place in the queue. (More information about this event cause, refer to section 5.17.4.1.5, “Event-cause RemainInQ vs. MultiAlert for Non-Group-device”).

5.17.4.2 Successful Group Call (Multiple Alerting with Parallel Ringing)

In this scenario device D1 calls a group of distribution mechanism (device D2) with members D3 and D4. There are devices available and the call is successfully distributed to devices D3 and D4 by the Group itself.



Activity	Monitored Device D1	Monitored Device H2 (HG)	Monitored Device D3	Monitored Device D4	Comments
1. D1 goes off-hook	**Service Initiated** • initiatedConnection D1C1 • localConnectionInfo initiated • cause normal • servicesPermitted				
2. D1 completes dialling HG-access-code (1234)	**Digits Dialled** • diallingConnection D1C1 • diallingDevice D1 • diallingSequence "1234" • localConnectionInfo initiated • cause normal				
	Originated • originatedConnection D1C1 • callingDevice D1 • calledDevice H2 • originatingDevice D1 • localConnectionInfo connected • cause normal • servicesPermitted				

Table 5-66 Successful Group Call (Multiple Alerting with Parallel Ringing) (page 1 of 3)

Activity	Monitored Device D1		Monitored Device H2 (HG)		Monitored Device D3		Monitored Device D4		Comments
3. The call reaches th HG	**Delivered** • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDev • localConnectionInfo • cause • servicesPermitted	 H2C1 H2 D1 H2 NS connected enterDist ClearConn SendU	**Delivered** • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDev • localConnectionInfo • cause • servicesPermitted	 H2C1 H2 D1 H2 NS alerting enterDist SendUI					
4. Call reaches the HG queue	**QueuedEvent** • connection • queuedDevice • callingDevice • calledDevice • lastRedirectionDev • localConnectionInfo • cause • servicesPermitted	 H2C1 H2 D1 H2 NS connected NoAvail Agents ClearConn SendUI	**QueuedEvent** • connection • queuedDevice • callingDevice • calledDevice • lastRedirectionDev • localConnectionInfo • cause • servicesPermitted	 H2C1 H2 D1 H2 NS queued NoAvail Agents ClearConn, SendUI					
5. The call begins to alert at D3 and D4	**DeliveredEvent** • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDev • localConnectionInfo • cause • servicesPermitted	 H2C1 H2 D1 H2 NS connected MultiAlert ClearConn SendUI	**DeliveredEvent** • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDev • localConnectionInfo • cause • servicesPermitted	 H2C1 H2 D1 H2 NS alerting MultiAlert SendUI	**DeliveredEvent** • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDev • localConnectionInfo • cause • servicesPermitted	 D3C1 D3 D1 H2 NS alerting MultiAlert Answer ClearConn SendUI	**DeliveredEvent** • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDev • localConnectionInfo • cause • servicesPermitted	 D4C1 D4 D1 H2 NS alerting MultiAlert Answer ClearConn SendUI	Please note: No Deflect is allowed for D3 because this is a MultiAlert-sitation

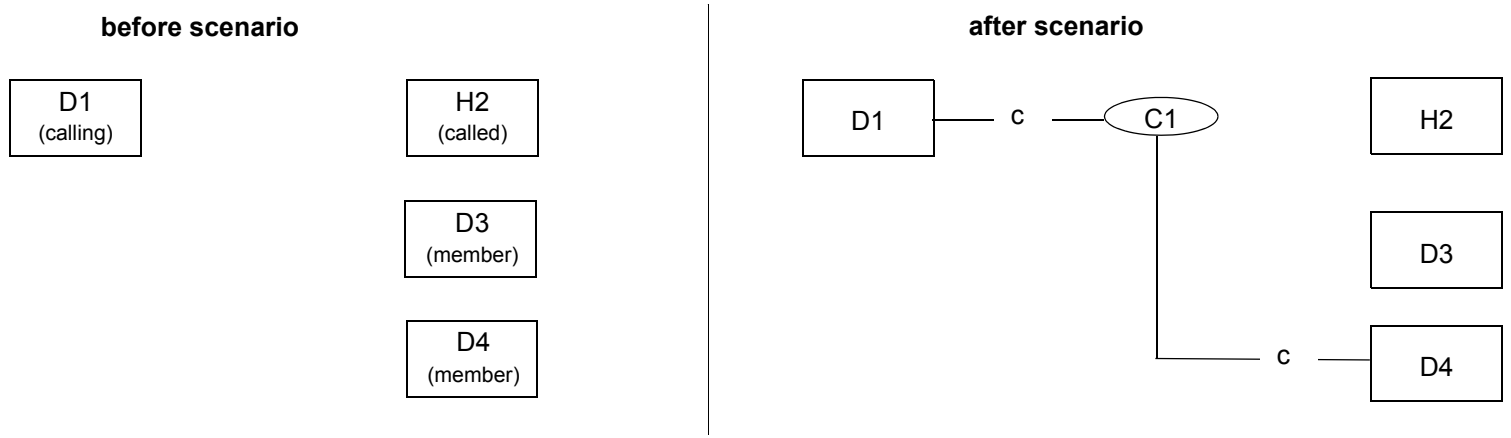
Table 5-66 Successful Group Call (Multiple Alerting with Parallel Ringing) (page 2 of 3)

Activity	Monitored Device D1	Monitored Device H2 (HG)	Monitored Device D3	Monitored Device D4	Comments
6. D4 answers call - HG-device leaves the call	**Connection Cleared** • droppedConnection H2C1 • releasingDevice H2 • localConnectionInfo connected • cause MultiAlert • servicesPermitted ClearConn Consult, Hold SST GenDg GenTelTon SendUI	**Connection Cleared** • droppedConnection H2C1 • releasingDevice H2 • localConnectionInfo null • cause MultiAlert • servicesPermitted none			
7. D4 is connected to the original call	**Established** • establishedConn D4C1 • answeringDevice D4 • callingDevice D1 • calledDevice H2 • lastRedirectionDev NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn Consult, Hold SST GenDg GenTelTon SendUI			**Established** • establishedConn D4C1 • answeringDevice D4 • callingDevice D1 • calledDevice H2 • lastRedirectionDev NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn Consult Hold SST GenDg GenTelTon SendUI	
8. Alerting connection cleard on D3.			**Connection Cleared** • droppedConnection D3C1 • releasingDevice D3 • localConnectionInfo Alerting • cause CallNotAnswered • servicesPermitted none **Connection Cleared** • droppedConnection D3C1 • releasingDevice D3 • localConnectionInfo null • cause normalClr • servicesPermitted none		

Table 5-66 Successful Group Call (Multiple Alerting with Parallel Ringing) (page 3 of 3)

5.17.4.3 Internal call to HG, Hunt Advance

D1 calls H2 (HG; H2 pilot-number). HG distributes the call to its member D3. D3 does not answer; HG performs a Hunt Advance to D4. D4 answers the call.



Activity	Monitored Device D1	Monitored Device H2 (HG)	Monitored Device D3	Monitored Device D4	Comments
1. D1 goes off-hook	**Service Initiated** • initiatedConnection D1C1 • initiatingDevice D1 • localConnectionInfo initiated • cause normal • servicesPermitted ClearConn, DialDg				

Table 5-67 Internal call to HG - Hunt-Advance (page 1 of 4)

Activity	Monitored Device D1	Monitored Device H2 (HG)	Monitored Device D3	Monitored Device D4	Comments
2. D1 completes dialling HG-access-code (1234)	**Digits Dialled** • diallingConnection D1C1 • diallingDevice D1 • diallingSequence "1234" • localConnectionInfo initiated • cause normal				
	Originated • originatedConnection D1C1 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn				
3. The call hits the Hunt-Group-Device	**Delivered** • connection H2C1 • alertingDevice H2 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo connected • cause enterDist • servicesPermitted ClearConn SendUI	**Delivered** • connection H2C1 • alertingDevice H2 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo alerting • cause enterDist • servicesPermitted SendUI			
4. HG distributes the call to HG-member D3	**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo connected • cause multiAlert • servicesPermitted ClearConn SendUI	**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo alerting • cause multiAlert • servicesPermitted SendUI	**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo alerting • cause multiAlert • servicesPermitted Answer ClearConn SendUI		**Please note:** No Deflect is allowed for D3 because this is a Multi-Alert-situation

Table 5-67 Internal call to HG - Hunt-Advance (page 2 of 4)

Activity	Monitored Device D1	Monitored Device H2 (HG)	Monitored Device D3	Monitored Device D4	Comments
5. D3 did not answer the call HG performs a Hunt Advance: D3 is cleared from the call.	**Connection Cleared** • droppedConnection D3C1 • releasingDevice D3 • localConnectionInfo connected • cause normalClr • servicesPermitted ClearConn SendUI	**Connection Cleared** • droppedConnection D3C1 • releasingDevice D3 • localConnectionInfo alerting • cause normalClr • servicesPermitted SendUI	**Connection Cleared** • droppedConnection D3C1 • releasingDevice D3 • localConnectionInfo null • cause normalClr • servicesPermitted none		
6. The next HG-member D4 is alerted	**Delivered** • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo connected • cause multiAlert • servicesPermitted ClearConn SendUI	**Delivered** • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo alerting • cause multiAlert • servicesPermitted ClearConn SendUI		**Delivered** • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo alerting • cause multiAlert • servicesPermitted Answer ClearConn SendUI	
7. D4 answers call - HG-device withdraws from the call	**Connection Cleared** • droppedConnection H2C1 • releasingDevice H2 • localConnectionInfo connected • cause multiAlert • servicesPermitted ClearConn Consult, Hold SST GenDg GenTelTon SendUI	**Connection Cleared** • droppedConnection H2C1 • releasingDevice H2 • localConnectionInfo null • cause multiAlert • servicesPermitted none		**Connection Cleared** • droppedConnection H2C1 • releasingDevice H2 • localConnectionInfo alerting • cause multiAlert • servicesPermitted ClearConn Consult Hold SST GenDg GenTelTon SendUI	HG withdraws from the call as soon as the member answers the call

Table 5-67 Internal call to HG - Hunt-Advance (page 3 of 4)

Activity	Monitored Device D1	Monitored Device H2 (HG)	Monitored Device D3	Monitored Device D4	Comments
8. D4 is connected to D1	**Established** • establishedConn D4C1 • answeringDevice D4 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn Consult, Hold SST GenDg GenTelTon SendUI			**Established** • establishedConn D4C1 • answeringDevice D4 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn Consult Hold SST GenDg GenTelTon SendUI	

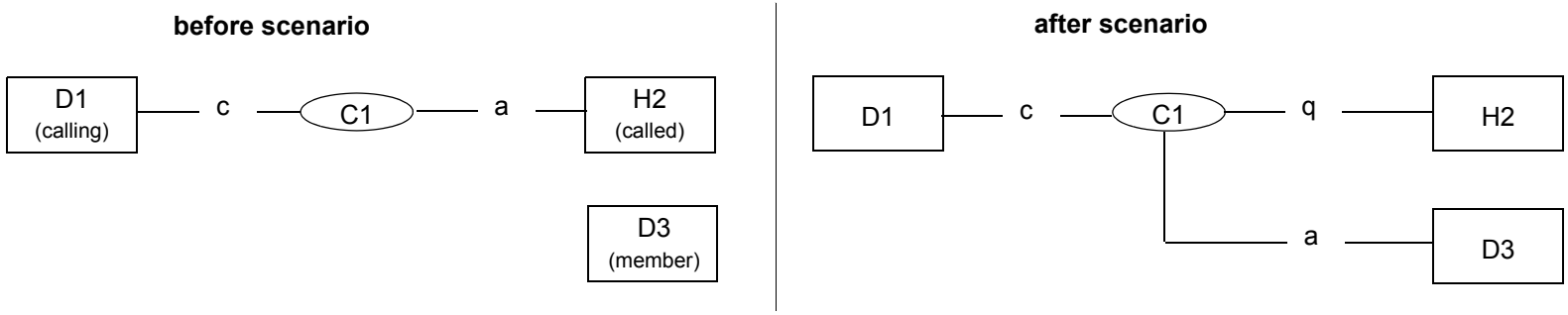
Table 5-67 Internal call to HG - Hunt-Advance (page 4 of 4)

Remark:

None

5.17.4.4 Call is queued at HG

D1 calls H2 (HG; H2 pilot-number). No members are available - the call is queued. Eventually HG-member D3 becomes available. HG distributes the call to D3.



Please refer to section 5.17.4.3, “Internal call to HG, Hunt Advance” for the event flow that leads to the „before“-state.

Activity	Monitored Device D1	Monitored Device H2 (HG)	Monitored Device D3 (member)	Comments
1. No member is available in the HG, the call is queued	**Queued** • queuedConnection H2C1 • queue H2 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo connected • cause NoAgents • servicesPermitted ClearConn SendUI	**Queued** • queuedConnection H2C1 • queue H2 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo queued • cause NoAgents • servicesPermitted **Deflect** SendUI		**Deflect:** This is the only situation where Deflect from a HG is possible
2. HG-member D3 becomes available; HG distributes the call to D3.	**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo connected • cause **remainsInQ** • servicesPermitted ClearConn SendUI	**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo queued • cause **remainsInQ** • servicesPermitted SendUI	**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo alerting • cause **remainsInQ** • servicesPermitted Answer ClearConn SendUI	**remainsInQ:** This cause is only provided if the HG is monitored. If HG is not monitored, event-cause „multiAlert“ will be provided instead.

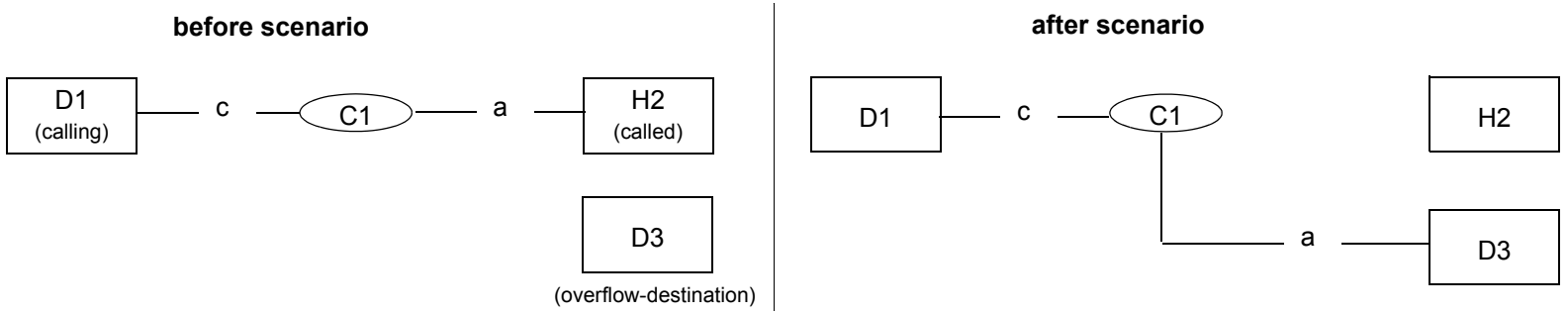
Table 5-68 Internal Call to HG – call is queued

Remark:

None

5.17.4.5 Call is routed to overflow-destination

D1 calls H2 (HG; H2 pilot-number). No member is available, no queue is configured for the HG. The call is redirected immediately to the overflow-destination.



Please refer to section 5.17.4.3, “Internal call to HG, Hunt Advance” for the event flow that leads to the „before“-state.

Activity	Monitored Device D1	Monitored Device H2 (HG)	Monitored Device D3	Comments
1. HG diverts the call to the overflow-destination		**Diverted** • connection H2C1 • divertingDevice H2 • newDestination D3 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo null • cause overflow • servicesPermitted none		
2. The overflow-destination is rung	**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev H2 • localConnectionInfo connected • cause overflow • servicesPermitted ClearConn CallBack SendUI		**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev H2 • localConnectionInfo alerting • cause overflow • servicesPermitted Answer ClearConn Deflect SendUI	

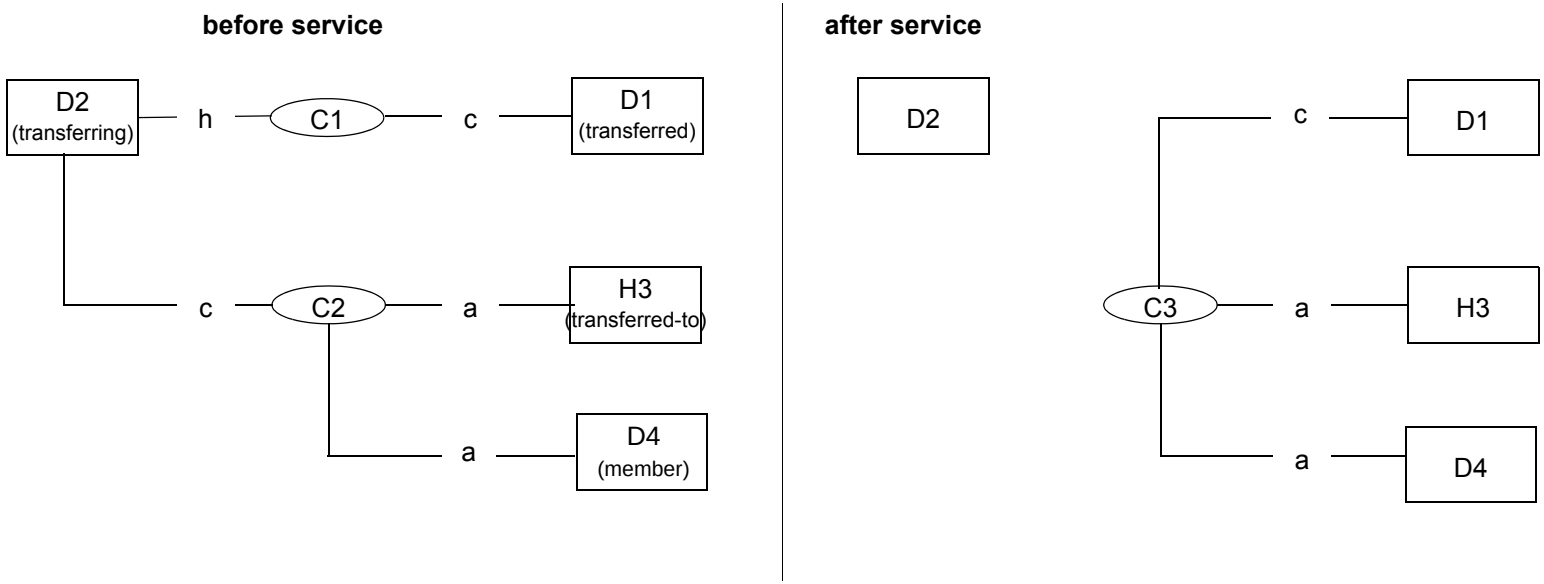
Table 5-69 Internal Call to HG – redirected to overflow-destination

Remark:

Please note that the call-scenario is different if a queue is configured for the HG, but the queue is full: in that case, HiPath 4000 treats the scenario as if the HG was forwarded immediately to the overflow-destination.

5.17.4.6 Transfer Ringing into HG

D1 is in a two-party conversation with D2 (Call-Id C1) and has initiated a consultation to H3 (HG; H3-pilot-number) (Call-Id C2), HG rings its member D4. While both H3 and D4 are ringing (Multi-Alert), D2 transfers the call (Call-Id C3).



Activity	Monitored Device D1		Monitored Device D2		Monitored Device H3 (HG)		Monitored Device D4 (member)		Comments
1. D2 transfers the call	**Transferred**		**Transferred**		**Transferred**		**Transferred**		**Conn-List:** For the HG (H3) and its member D4, both H3 and D4 are shown as old Conn-Ids in the list, because they belong to the same call
	• primaryOldCall	D1C1	• primaryOldCall	D2C1	• primaryOldCall	H3C2	• primaryOldCall	D4C2	
			• secondaryOldCall	D2C2	•		•		
	• transferringDevice	D2	• transferringDevice	D2	• transferringDevice	D2	• transferringDevice	D2	
	• transferredDevice	H3	• transferredDevice	H3	• transferredDevice	H3	• transferredDevice	H3D3	
	• TransferConnList		• TransferConnList:		• TransferConnList:		• TransferConnList:		
	1.new / old	(D1C3) / (D1C1)	1.new / old	(D1C3) / (D1C1)	1.new	(D1C3)	1.new	(D1C3)	
	2. new	(D4C3)	2. new / old	(D4C3) / (D4C2)	2. new / old	(D4C3) / **(D4C2)**	2. new / old	(D4C3) / (D4C2)	
	3. new	(H3C3)	3. new / old	(H3C3) / (H3C2)	3. new / old	(H3C3) / (H3C2)	3. new / old	(H3C3) / **(H3C2)**	
	• localConnectionInfo	connected	• localConnectionInfo	null	• localConnectionInfo	alerting	• localConnectionInfo	alerting	
	• cause	Transfer	• cause	Transfer	• cause	Transfer	• cause	Transfer	
	• servicesPermitted	ClearConn, SendUI	• servicesPermitted	none	• servicesPermitted	SendUI	• servicesPermitted	Answer ClearConn SendUI	

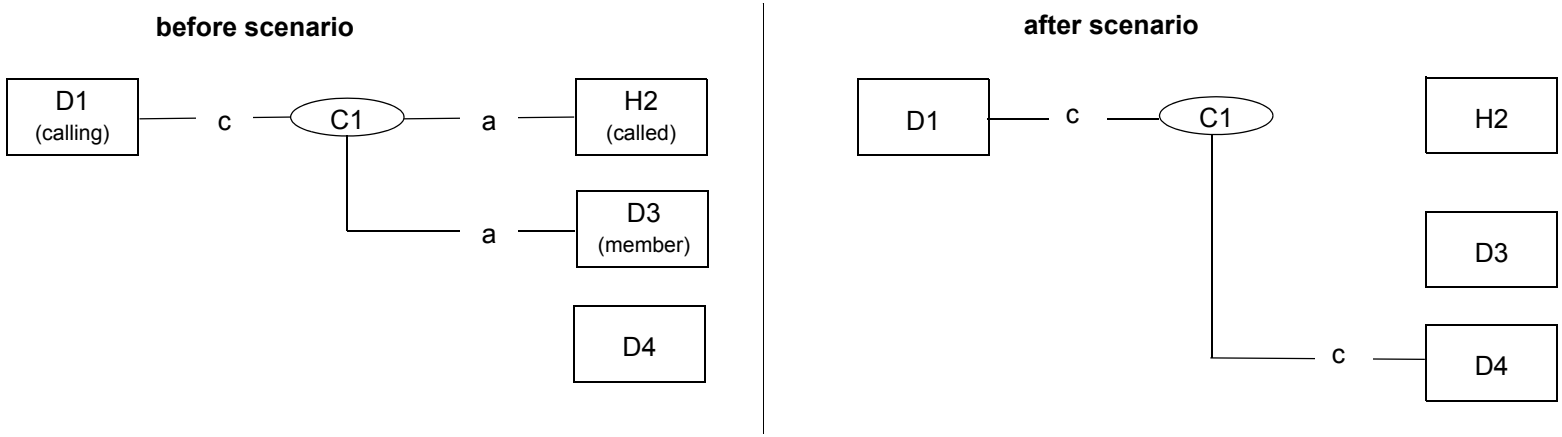
Table 5-70 Consultation to HG - Transfer ringing

Remark:

If D4 does not answer the call, HiPath 4000 will **not** perform a Transfer-Recall as would be the case in a Transfer-scenario without a Group-Device.

5.17.4.7 Pick from HG-member

D1 calls H2 (HG; H2 pilot-number), HG distributes the call to its member D3. While D3 is ringing, D4 picks the call.



Please refer to section 5.17.4.3, “Internal call to HG, Hunt Advance” for the event flow that leads to the „before“-state.

Activity	Monitored Device D1	Monitored Device H2 (HG)	Monitored Device D3 (member)	Monitored Device D4	Comments
1. D4 picks call from D3 - First, the Hunt-Group-Device leaves the call	**Connection Cleared** • droppedConnection H2C1 • releasingDevice H2 • localConnectionInfo connected • cause normalClr • servicesPermitted ClearConn Consult, Hold SST GenDg GenTelTon SendUI	**Connection Cleared** • droppedConnection H2C1 • releasingDevice H2 • localConnectionInfo null • cause normalClr • servicesPermitted none	**Connection Cleared** • droppedConnection H2C1 • releasingDevice H2 • localConnectionInfo alerting • cause normalClr • servicesPermitted none		

Table 5-71 Pick from HG-member(page 1 of 2)

Activity	Monitored Device D1	Monitored Device H2 (HG)	Monitored Device D3 (member)	Monitored Device D4	Comments
2. The call is diverted from member D3 to D4			**Diverted** • connection D3C1 • divertingDevice D3 • newDestination D4 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev NS • localConnectionInfo null • cause pick • servicesPermitted none		
3. D4 is connected to D1	**Established** • establishedConn D4C1 • answeringDevice D4 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev D3 • localConnectionInfo connected • cause pick • servicesPermitted ClearConn Consult, Hold SST GenDg GenTelTon SendUI			**Established** • establishedConn D4C1 • answeringDevice D4 • callingDevice D1 • calledDevice H2 pilot • lastRedirectionDev D3 • localConnectionInfo connected • cause pick • servicesPermitted ClearConn Consult, Hold SST GenDg GenTelTon SendUI	

Table 5-71 Pick from HG-member(page 2 of 2)

Remark:

None

5.17.5 General Attendant (GA)

5.17.5.1 General description GA

5.17.5.1.1 Introduction

Monitoring of General Attendant (former „Attendant Queue“) was provided in a non-standard way in previous versions. Please note that the call-flow has changed considerably!

CSTA III (ECMA 269) and the ECMA Call-Scenarios do not provide much information about how devices like GA should be modeled. Therefore it is necessary to describe the model CA 4000 uses.

5.17.5.1.2 Characteristics of GA

The basic characteristics for GA are the same as for the HG. Please refer to section 5.17.4.1.2, “Characteristics of HG” for more information. The mask used for monitoring is 0x05000000.

5.17.5.1.3 Deflect and GA

The following has to be considered when using the Deflect-Call-Request for calls where GA is involved:

- Deflect is never allowed from the GA
- Deflect-Call is **not** allowed in a Multiple-Alert-Situation (this includes **all** devices involved in the call).

5.17.5.1.4 Different types of GA

There are two different types of GA. They differ in how the calls are distributed to their members (Attendant Console or ACs) and the number of queues provided:

- GA2Q
- GAMQ

5.17.5.1.4.1 GA2Q

2Q means „Double Queue“. It provides 2 queues:

- for external calls (German: „Amt“)
- for internal calls (German: „Melde“)

The distributing mechanism is similar to the Hunt-Groups - please refer to section 5.17.4.1, “General description HG” for more details.

5.17.5.1.4.2 GAMQ

MQ means „Multi Queue“. This GA can have up to 12 queues configured. It depends upon Hi-Path 4000-configuration, which calls are placed in which queue.

The calls are not distributed by GAMQ, but every call is queued. The ACs pick the calls from the queue. This results in a different CSTA-event-flow when compared with GA2Q:

- Because all calls are queued in GAMQ, the event-cause for the Queued-Event is always „normal“ (not „noAvailAgents“ like for GA2Q or HG).
- Diverted-Event for GA when AC picks up call (event-cause = distributed) (the AC is a new destination for the call, therefore a Diverted-Event).
- Delayed Delivered-Event for GA after picking up (with LastRedir = GA), immediately followed by an Established-Event

Please refer to section 5.17.5.2, “Internal call to GA2Q” for more information.

5.17.5.1.5 Special Features of GA

The following features are similar to HG and therefore not described in detail:

- **Members**: the members of a GA are Attendant Consoles (AC or ATC)
- **Control of the call**: like for HG, the GA remains in control of the call until the call is either successfully distributed or torn down. One exception is night-service: there are certain types of night-service where GA leaves the call as soon as the night-destination starts ringing (for more details refer to section 5.17.5.1.5, “Night-service”).
- **Queues**: A GA usually has more than one queues. A Deflect from the GA-Queue is never permitted. (Refer to sections 5.17.5.1.4, “GA2Q” and 5.17.5.1.4, “GAMQ” for more information)

The following features are special GA-features:

5.17.5.1.5.1 Intercept

Intercept is an important feature for the GA. Here a few examples when intercept may occur:

- A calls B, B does not answer; the call is intercepted to GA
  - without parallel call: B stops ringing (practically a diversion of the call)
  - with parallel call: after the call has been intercepted to GA, B keeps ringing. If it is an GA2Q, 3 devices may be ringing at the same time: B, GA and AC. Whoever answers the call first (B or AC) is connected to the call, all other connections are cleared.
- A calls B, B is busy; the call is intercepted to GA
- A dials an invalid extension, an incomplete extension or no extension at all; the call is intercepted to GA.
  Please note: the dialled-digits will be shown as CalledDevice in all subsequent events. If no digits were dialled (no extension), CallBridge will report „NotKnown“.

When and if a call is intercepted depends upon HiPath 4000-configuration. Because CSTA III does not provide an appropriate event-cause for intercept, an application must infer from the event-flow that intercept has occured. (See example-event-flows in sections 5.17.5.5 to 5.17.5.8).

5.17.5.1.5.2 Night-service

After the last AC goes out-of-service, the GA switches to night-mode. There are different kinds of night-service (the desired kind of night-service can be configured on the switch):

- **Centralized attendant internal (ZVFINT)**
  The GA (GA1) is forwarded to another GA (GA2) in the same node. GA1 leaves the call as soon as the call is diverted to GA2.
- **Centralized attendant external (ZVFEXT)**
  The GA (GA1) is forwarded to another GA (GA2) in another node. GA1 remains involved in the call until an AC on GA2 answers the call.
- **Local night station(s)**
  The GA forwards incoming calls to nightstations (DIGITEs, ANATEs) in the same node. If all night stations are busy, the call remains in the queue of the GA. As soon as the night-station starts ringing, the GA is not involved in the call any more. This is different to day-service, when GA remains involved until the AC answers the call
- **Universal night answer („Allgemeines Abfragen“)**
  A special UNA-device (e.g. a bell) is configured, to which all incoming calls are forwarded. Subscribers (e.g. DIGITEs) can pick up calls from the UNA-device. The UNA-device is a so-called „specialDevice“ - this device is represented by a device-number with the mask 0x0a000000. Please note: this specialDevice cannot be monitored!. GA leaves the call as soon as the UNA-device starts ringing.
- **No destination configured („Leervariante“)**
  Incoming calls are queued in GA and remain queued until they either hang up, or an AC becomes available (day service is started)
- **Trunk Night Service (TNS)**
  A specific trunk has its own night destination configured. An incoming call is forwarded to this destination (on the same node or another node). In this case, the GA is only involved in the call if the individual TNS destination is busy.

5.17.5.1.5.3 Recalls to GA

There are different reasons why a recall to GA occurs:

- **Serial Recall:**
  Example: A and B are connected, the connection was formerly established by an AC using the serial call feature. When B goes onhook, A (=calling party) seizes the GA immediately again.

- **Trunk to trunk supervision**
  Example: If two trunks without disconnect supervision are connected, HiPath 4000 cannot clear this call properly. Therefore HiPath 4000 starts a timer. Whenever this timer expires, a recall to the GA is started. The AC listens in on the call and decides whether to disconnect the call or not.
- **Transfer Recall**:
  Example: A calls GA, an AC transfers the call to B. B does not answer - the GA is recalled.
- **Park Recall**:
  Example: An AC parks a call with B (directed call park). The Park-Timer expires, B starts ringing. The Recall-Timer expires - the GA is recalled

5.17.5.1.5.4 Personal calls

All personal calls to the AC are handled via the GA. In previous switch-versions, personal calls to the AC did not involve the group-device.

5.17.5.1.5.5 De-Queueing of calls

GA is capable of removing calls from its queue without tearing down the call. Whenever GA dequeues a call, the calling party is in the state „Waits for place in queue“. This state is similar to the originated state. A Conn-Cleared-Event is sent for the monitor of GA and the calling device to show this situation.

When the call is finally accepted by GA, a Delivered-Event is sent. OpenScape 4000 has lost information of the history of the call - it will be shown as if the call has directly dialled the GA (no LastRedirectionDevice, no special event-cause, etc.).

5.17.5.1.5.6 Diversion to GA fails

When all ACs are busy, no place is available in the queue and no overflow-destination is configured, GA rejects calls that are diverted. After the rejection of GA, the call is in the state „Waits for place in queue“.

Here is a list of some situations where this might happen:

- A ringing at B, recall to GA, no parallel call, GA rejects call
- A ringing at B, CF-NA to GA, no parallel call, GA rejects call
- A ringing at B, Intercept to GA, no parallel call, GA rejects call

In these cases, CA 4000 sends a Diverted-Event for the diverting party (B). No event is generated for the calling party (A). The next event the calling party will receive is the Delivered-Event when the call is finally accepted by GA (in this special case, LastRedirectionDevice will be NS because the history of the call is no longer available for HiPath 4000).

5.17.5.1.5.7 Speed extend from Attendant Console

Attendant console is able to speed transfer a connected call. The event flow is modelled like a single step transfer, without request and response involved. There are no events showing the speed dial of the destination. Tis is valid also for Attendant Console Light. In case of unsuccessful speed extend the call remains as it was before, no events will be provided unless the call was offered to the destination. The call flow for the speed extend to a busy destination with offered mode activated is the same as the single step transfer attempt to a busy destination with offered mode activated.

5.17.5.1.6 Known Restrictions for GA

The restrictions are the same as for HG, please refer to section 5.17.4.1.5, “Known Restrictions for HG”.

5.17.5.1.7 General Rules concerning Multi-Alert-Situations for GA

The same general rules as for HG apply for GA (please refer to section 5.17.5.1.7, “General Rules concerning Multi-Alert-Situations for GA”). Additional situations where Multi-Alert may occur for GA are:

- Intercept with parallel call to the GA. Both the intercepted device and the GA are ringing. If GA distributes the call to one of its members, 3 devices are ringing at the same time. (see section 5.17.5.6, “Intercept with parallel call to GA2Q”)
- Recall to the GA with parallel call: a call that has been transferred by an AC is not answered => a recall to the GA is performed

Additional rules for GA:

1. Whenever a device leaves a call and the call remains in a Multiple-Alerting-Situation, a Conn-Cleared-Event must be sent.

   Example:

   - Intercept with parallel call, three devices are ringing simultaneously (B, GA and AC), AC goes out-of-service, therefore the AC is dropped from the call

2. If a device leaves a call, leaving the call in Multiple-Alerting-Situation and another device joins the call in one step (e.g. overflow from one GA to another GA during Multiple-Alerting), this is **not** shown as a Diversion, but as one device leaving the call (Conn-Cleared) and another device added to the call (Delivered-Event with LastRedirection=NS and event-cause=MultiAlert). It would be wrong to send a Diverted-Event, because this would violate the rule that a diversion has **not** taken place when event-cause is Multi-Alert (see rule # 2 in section 5.17.4.1.6, “General Rules concerning Multi-Alert-Situations”). The remaining devices need to be informed that a device has left the call - this is done by sending a Connection-Cleared event to all remaining devices.

   Example:

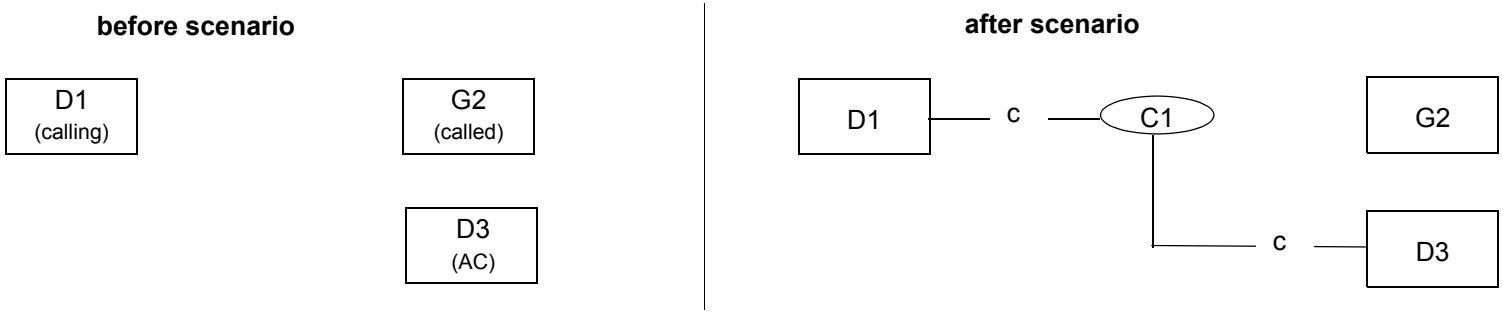
- A calls B, call is intercepted to GA1 with parallel call, no AC is free, the call is queued at GA1. After an overflow timer expires, the call is diverted to GA2. In this case, a Conn-Cleared-Event is sent for GA1 and Delivered-Event for GA2, but no Diverted-Event (see section 5.17.5.8, “Intercept with parallel call, call is routed to another GA after timeout”)

5.17.5.2 Internal call to GA2Q

The scenario is identical to the HG-scenario; please refer to Section 5.17.4.3, “Internal call to HG, Hunt Advance” for more information (please note that a Hunt-Advance as shown in this scenario is usually not performed at a GA-device).

5.17.5.3 Internal call to GAMQ

D1 calls G2 (GAMQ, internal attendant access code: G2-int), the call is queued. D3 (AC) picks the call from the queue.



Activity	Monitored Device D1		Monitored Device G2 (GAMQ)	Monitored Device D3 (AC)	Comments
1. D1 goes off-hook.	**Service Initiated**				
	• initiatedConnection	D1C1			
	• initiatingDevice	D1			
	• localConnectionInfo	initiated			
	• cause	normal			
	• servicesPermitted	ClearConn DialDg			
2. D1 completes dialling the internal attendant access code..(1234)	**Digits Dialled**				
	• diallingConnection	D1C1			
	• diallingDevice	D1			
	• diallingSequence	“1234“			
	• localConnectionInfo	initiated			
	• cause	normal			
	• servicesPermitted	none			
	Originated				
	• originatedConnection	D1C1			
	• callingDevice	D1			
	• calledDevice	G2-int			
	• lastRedirectionDev	NS			
	• localConnectionInfo	connected			
	• cause	normal			
	• servicesPermitted	ClearConn			

Table 5-72 Internal Call to GAMQ (page 1 of 3)

Activity	Monitored Device D1	Monitored Device G2 (GAMQ)	Monitored Device D3 (AC)	Comments
3. The call hits the GAMQ .	**Delivered** • connection G2C1 • alertingDevice G2 • callingDevice D1 • calledDevice G2-int • lastRedirectionDev NS • localConnectionInfo connected • cause enterDist • servicesPermitted ClearConn SendUI	**Delivered** • connection G2C1 • alertingDevice G2 • callingDevice D1 • calledDevice G2-int • lastRedirectionDev NS • localConnectionInfo alerting • cause enterDist • servicesPermitted SendUserInfo		
4. The call is queued at GAMQ **Please note:** all calls are queued at GAMQ. GAMQ does not distribute calls, but the ACs pick the calls from the queue	**Queued** • queuedConnection G2C1 • queue G2 • callingDevice D1 • calledDevice G2-int • lastRedirectionDev NS • localConnectionInfo connected • cause **normal** • servicesPermitted ClearConn SendUI	**Queued** • queuedConnection G2C1 • queue G2 • callingDevice D1 • calledDevice G2-int • lastRedirectionDev NS • localConnectionInfo queued • cause **normal** • servicesPermitted SendUI		**cause = normal:** because all calls are queued at GAMQ, event-cause is „normal“ instead of „noAgents“
5. AC picks the call from the queue - first, the call is diverted from the GAMQ		**Diverted** • connection G2C1 • divertingDevice G2 • newDestination D3 • callingDevice D1 • calledDevice G2-int • lastRedirectionDev NS • localConnectionInfo null • cause distributed • servicesPermitted none		

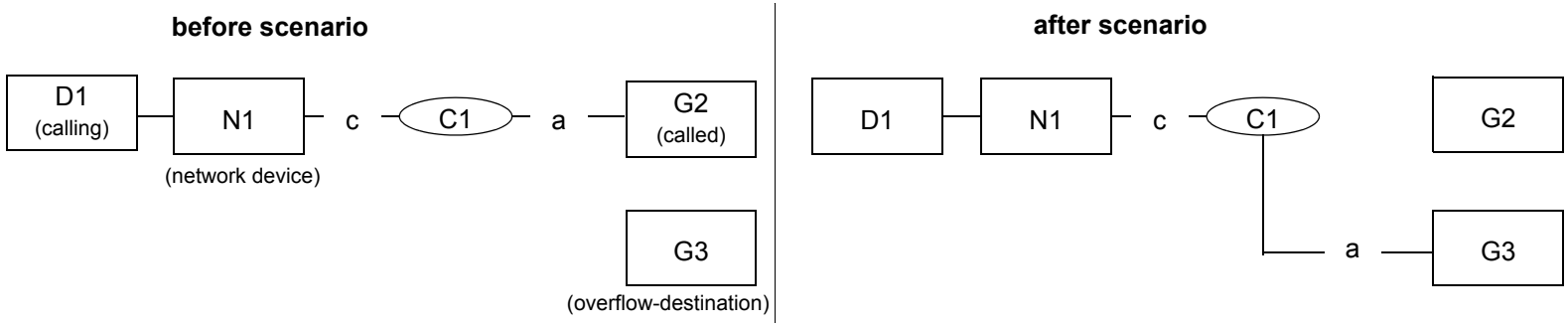
Table 5-72 Internal Call to GAMQ (page 2 of 3)

Activity	Monitored Device D1	Monitored Device G2 (GAMQ)	Monitored Device D3 (AC)	Comments
6. The call arrives at the AC.	**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice G2-int • lastRedirectionDev G2 • localConnectionInfo connected • cause distributed • servicesPermitted ClearConn Consult Hold SST GenDg GenTelTon SendUI		**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice G2-int • lastRedirectionDev G2 • localConnectionInfo alerting • cause distributed • servicesPermitted SendUI	A delayed Delivered-Event is sent for the AC after the AC picks the call.
7. The call is established.	**Established** • establishedConn D3C1 • answeringDevice D3 • callingDevice D1 • calledDevice G2-int • lastRedirectionDev G2 • localConnectionInfo connected • cause normal • servicesPermitted ClearConn Consult Hold SST GenDg GenTelTon SendUI		**Established** • establishedConn D3C1 • answeringDevice D3 • callingDevice D1 • calledDevice G2-int • lastRedirectionDev G2 • localConnectionInfo connected • cause normal • servicesPermitted SendUI	

Table 5-72 Internal Call to GAMQ (page 3 of 3)

5.17.5.4 Overflow from one GA to another GA

External caller D1 calls G2 (GAMQ, external attendant access code: G2-ext). The queue of GAMQ is full, G3 (GA2Q) is configured as overflow destination. The call is immediately forwarded to GA2Q (before it is queued at the GAMQ).



Please refer to section 5.17.5.3, “Internal call to GAMQ” for the event flow that leads to the „before“-state.

Activity	Monitored Device N1	Monitored Device G2 (GA1)	Monitored Device G3 (GA2)	Comments
1. The queue of D2 is full, the call is diverted immediately to the overflow destination D3		**Diverted** • connection G2C1 • divertingDevice G2 • newDestination D3 • callingDevice D1 • calledDevice G2-ext • AssCallingDevice N1 • NWCallingDevice D1 • lastRedirectionDev NS • localConnectionInfo null • cause overflow • servicesPermitted none		

Table 5-73 Overflow from one GA to another GA (page 1 of 2)

Activity	Monitored Device N1		Monitored Device G2 (GA1)	Monitored Device G3 (GA2)		Comments
2. Overflow-destination D3 is rung	**Delivered**			**Delivered**		
	• connection	G3C1		• connection	G3C1	
	• alertingDevice	G3		• alertingDevice	G3	
	• callingDevice	D1		• callingDevice	D1	
	• calledDevice	G2-ext		• calledDevice	G2-ext	
	• OrigNIDConn	N1C1		• OrigNIDConn	N1C1	
	• NWCallingDevice	D1		• NWCallingDevice	D1	
	• AssCallingDevice	N1		• AssCallingDevice	N1	
	• lastRedirectionDev	G2		• lastRedirectionDev	G2	
	• localConnectionInfo	connected		• localConnectionInfo	alerting	
	• cause	overflow		• cause	overflow	
	• servicesPermitted	ClearCall SendUI		• servicesPermitted	SendUI	

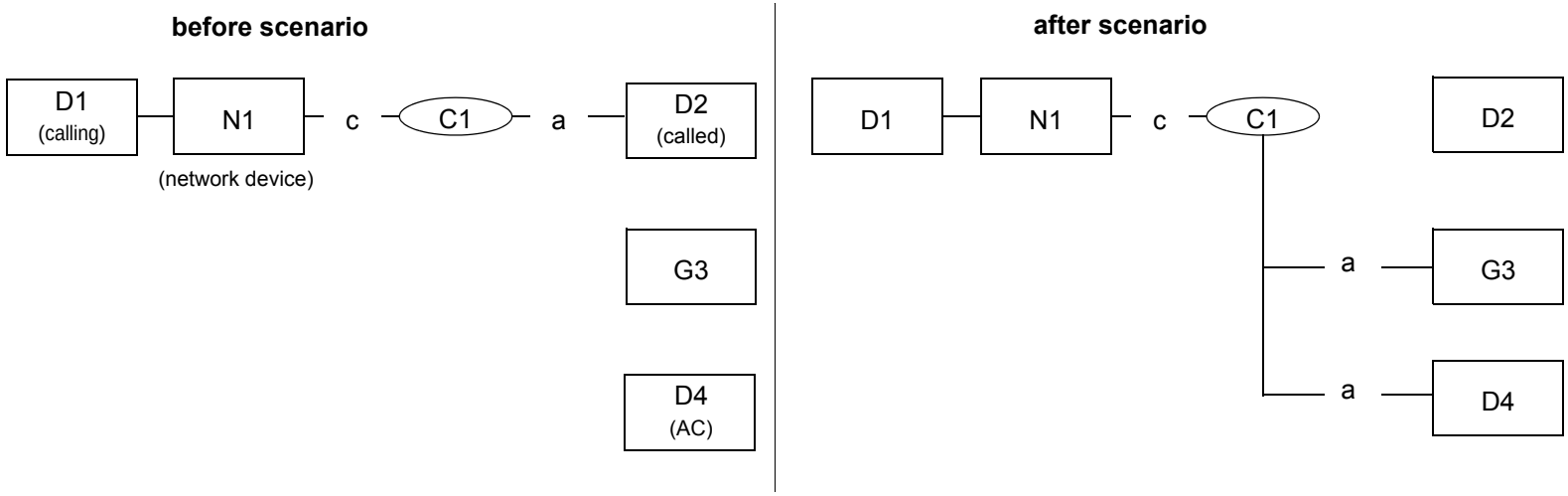
Table 5-73 Overflow from one GA to another GA (page 2 of 2)

Remark:

None

5.17.5.5 Intercept without parallel call to GA2Q

An external caller (directory-number D1) calls D2 via the network-interface N1 (trunk). D2 does not answer the call. After a timeout, the call is intercepted without parallel call to the GA (G3). GA distributes the call to the AC D4.



Activity	Monitored Device N1	Monitored Device D2	Monitored Device G3 (GA)	Monitored Device D4 (AC)	Comments
1. The call was previously ringing at D2. Intercept-timer has expired, the call is intercepted without parallel call to the GA		**Diverted** • connection D2C1 • divertingDevice D2 • newDestination G3 • calledDevice D2 • lastRedirectionDev NS • localConnectionInfo null • cause callNotAnswered • servicesPermitted none			

Table 5-74 Intercept without parallel call to GA2Q (page 1 of 2)

Activity	Monitored Device N1	Monitored Device D2	Monitored Device G3 (GA)	Monitored Device D4 (AC)	Comments
2. The GA is alerted	**Delivered** • connection G3C1 • alertingDevice G3 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev D2 • localConnectionInfo connected • cause enterDist • servicesPermitted ClearCall SendUI		**Delivered** • connection G3C1 • alertingDevice G3 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev D2 • localConnectionInfo alerting • cause enterDist • servicesPermitted SendUI		
3. GA2Q distributes call to AC	**Delivered** • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo connected • cause multiAlert • servicesPermitted ClearConn SendUI		**Delivered** • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo alerting • cause multiAlert • servicesPermitted SendUI	**Delivered** • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo alerting • cause multiAlert • servicesPermitted SendUI	

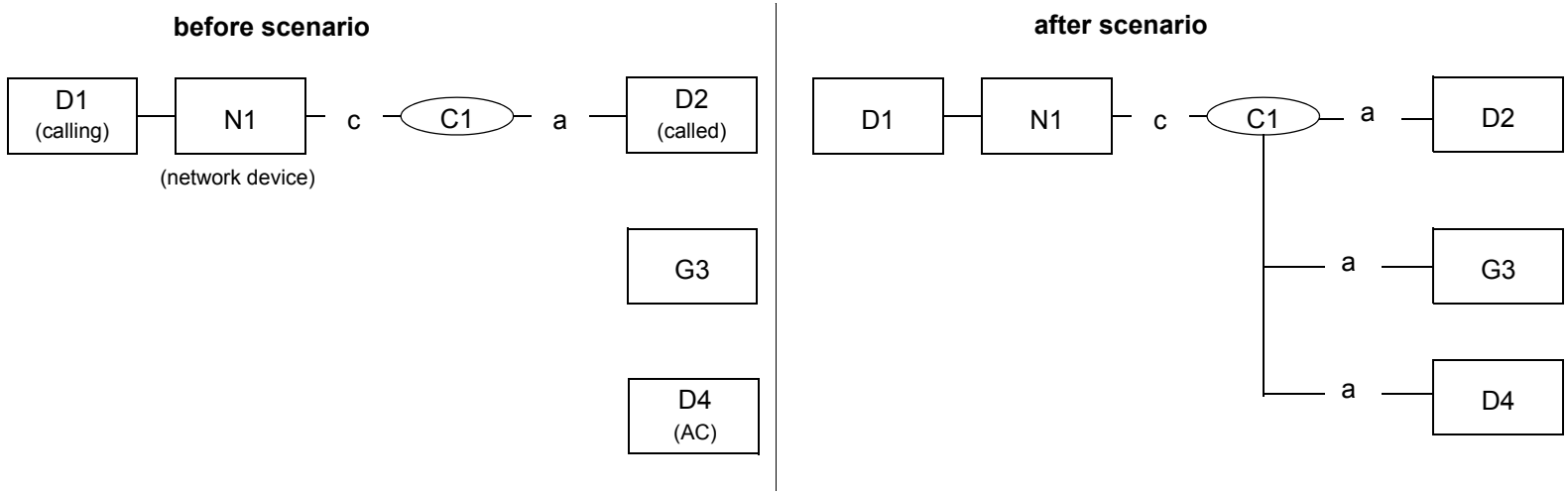
Table 5-74 Intercept without parallel call to GA2Q (page 2 of 2)

Remark:

None

5.17.5.6 Intercept with parallel call to GA2Q

An external caller (directory-number D1) calls D2 via the network-interface N1 (trunk). D2 does not answer the call. After a timeout, the call is intercepted with parallel call to the GA (D3). GA distributes the call to AC D4.



Activity	Monitored Device N1	Monitored Device D2	Monitored Device G3 (GA)	Monitored Device D4 (AC)	Comments
1. The call was previously ringing at D2 Intercept-timer has expired, the call is intercepted to GA (with parallel call), the GA is alerted	**Delivered** • connection G3C1 • alertingDevice G3 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo connected • cause multiAlert • servicesPermitted ClearCall SendUI	**Delivered** • connection G3C1 • alertingDevice G3 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo alerting • cause multiAlert • servicesPermitted Answer ClearCall SendUI	**Delivered** • connection G3C1 • alertingDevice G3 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo alerting • cause multiAlert • servicesPermitted SendUI		

Table 5-75 Intercept with parallel call to GA2Q (page 1 of 2)

Activity	Monitored Device N1		Monitored Device D2		Monitored Device G3 (GA)		Monitored Device D4 (AC)		Comments
2. GA2Q distributes call to AC	**Delivered**		**Delivered**		**Delivered**		**Delivered**		
	• connection	D4C1	• connection	D4C1	• connection	D4C1	• connection	D4C1	
	• alertingDevice	D4	• alertingDevice	D4	• alertingDevice	D4	• alertingDevice	D4	
	• callingDevice	D1	• callingDevice	D1	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	• calledDevice	D2	• calledDevice	D2	
	• OrigNIDConn	N1C1	• OrigNIDConn	N1C1	• OrigNIDConn	N1C1	• OrigNIDConn	N1C1	
	• NWCallingDevice	D1	• NWCallingDevice	D1	• NWCallingDevice	D1	• NWCallingDevice	D1	
	• AssCallingDevice	N1	• AssCallingDevice	N1	• AssCallingDevice	N1	• AssCallingDevice	N1	
	• lastRedirectionDev	NS	• lastRedirectionDev	NS	• lastRedirectionDev	NS	• lastRedirectionDev	NS	
	• localConnectionInfo	connected	• localConnectionInfo	alerting	• localConnectionInfo	alerting	• localConnectionInfo	alerting	
	• cause	multiAlert	• cause	multiAlert	• cause	multiAlert	• cause	multiAlert	
	• servicesPermitted	ClearConn SendUI	• servicesPermitted	Answer ClearCall SendUI	• servicesPermitted	SendUI	• servicesPermitted	SendUI	

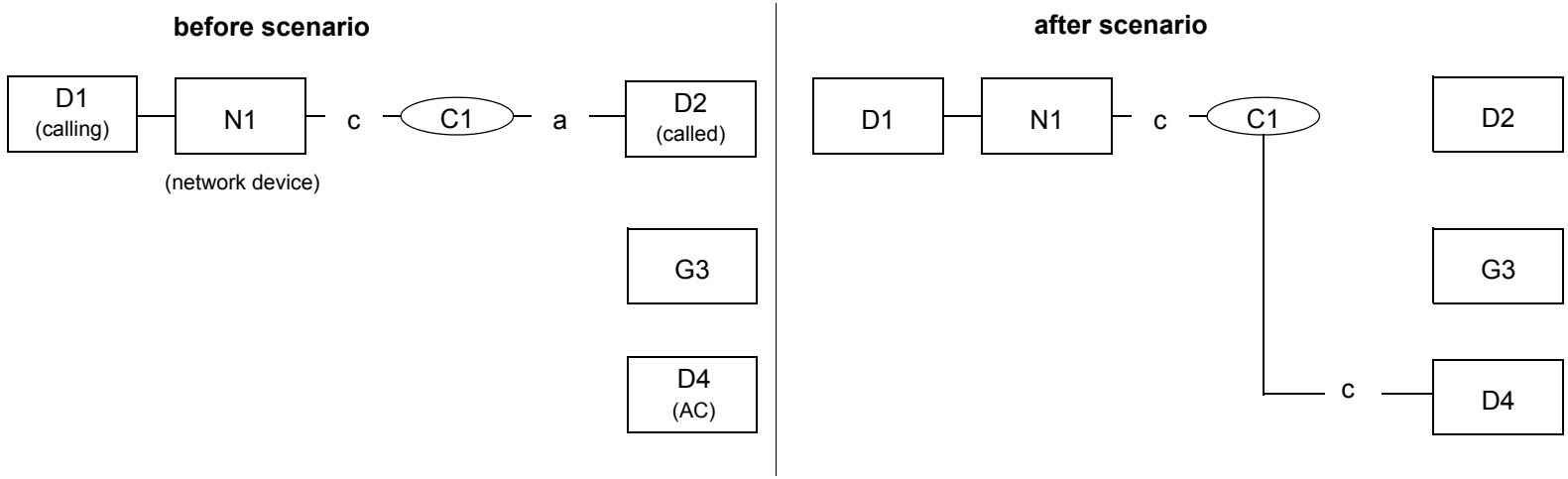
Table 5-75 Intercept with parallel call to GA2Q (page 2 of 2)

Remark:

Immediately after the AC has answered the call, it automatically seizes D2 again in a consultation-call (a new call-id will be created).

5.17.5.7 Intercept with parallel call to GAMQ

An external caller (directory-number D1) calls D2 via the network-interface N1 (trunk). D2 does not answer the call. After a timeout, the call is intercepted with parallel call to the GA (G3). The Attendant Console D4 answers the call.



Activity	Monitored Device N1		Monitored Device D2		Monitored Device G3 (GA)		Monitored Device D4 (AC)	Comments
1. The call was prevoiusly ringing at D2. Intercept-timer has expired, the call is intercepted to GAMQ (with parallel call), the GAMQ is alerted	**Delivered**		**Delivered**		**Delivered**			
	• connection	G3C1	• connection	G3C1	• connection	G3C1		
	• alertingDevice	G3	• alertingDevice	G3	• alertingDevice	G3		
	• callingDevice	D1	• callingDevice	D1	• callingDevice	D1		
	• calledDevice	D2	• calledDevice	D2	• calledDevice	D2		
	• OrigNIDConn	N1C1	• OrigNIDConn	N1C1	• OrigNIDConn	N1C1		
	• NWCallingDevice	D1	• NWCallingDevice	D1	• NWCallingDevice	D1		
	• AssCallingDevice	N1	• AssCallingDevice	N1	• AssCallingDevice	N1		
	• lastRedirectionDev	NS	• lastRedirectionDev	NS	• lastRedirectionDev	NS		
	• localConnectionInfo	connected	• localConnectionInfo	alerting	• localConnectionInfo	alerting		
	• cause	multiAlert	• cause	multiAlert	• cause	multiAlert		
	• servicesPermitted	ClearCall SendUI	• servicesPermitted	Answer ClearCall SendUI	• servicesPermitted	SendUI		

Table 5-76 Intercept with parallel call to GAMQ (page 1 of 3)

Activity	Monitored Device N1	Monitored Device D2	Monitored Device G3 (GA)	Monitored Device D4 (AC)	Comments
2. Call is queued at GAMQ	**Queued** • queuedConnection G3C1 • queue G3 • callingDevice D1 • calledDevice D2 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn SendUI	**Queued** • queuedConnection G3C1 • queue G3 • callingDevice D1 • calledDevice D2 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo alerting • cause normal • servicesPermitted Answer ClearConn SendUI	**Queued** • queuedConnection G3C1 • queue G3 • callingDevice D1 • calledDevice D2 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo queued • cause normal • servicesPermitted SendUI		
3. AC picks - D2 leaves the call	**Connection Cleared** • droppedConnection D2C1 • releasingDevice D2 • localConnectionInfo connected • cause normalClr • servicesPermitted ClearConn SendUI	**Connection Cleared** • droppedConnection D2C1 • releasingDevice D2 • localConnectionInfo null • cause normalClr • servicesPermitted none	**Connection Cleared** • droppedConnection D2C1 • releasingDevice D2 • localConnectionInfo queued • cause normalClr • servicesPermitted SendUI		
4. The call is diverted from GAMQ			**Diverted** • connection G3C1 • divertingDevice G3 • newDestination D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDev NS • localConnectionInfo null • cause distributed • servicesPermitted none		

Table 5-76 Intercept with parallel call to GAMQ (page 2 of 3)

Activity	Monitored Device N1	Monitored Device D2	Monitored Device G3 (GA)	Monitored Device D4 (AC)	Comments
5. The call arrives at the AC.	**Delivered** • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev G3 • localConnectionInfo connected • cause distributed • servicesPermitted ClearConn SendUI			**Delivered** • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev G3 • localConnectionInfo alerting • cause distributed • servicesPermitted SendUI	
6. The call is established	**Established** • establishedConn D4C1 • answeringDevice D4 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev G3 • localConnectionInfo connected • cause normal • servicesPermitted ClearConn SendUI			**Established** • establishedConn D4C1 • answeringDevice D4 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev G3 • localConnectionInfo connected • cause normal • servicesPermitted SendUI	

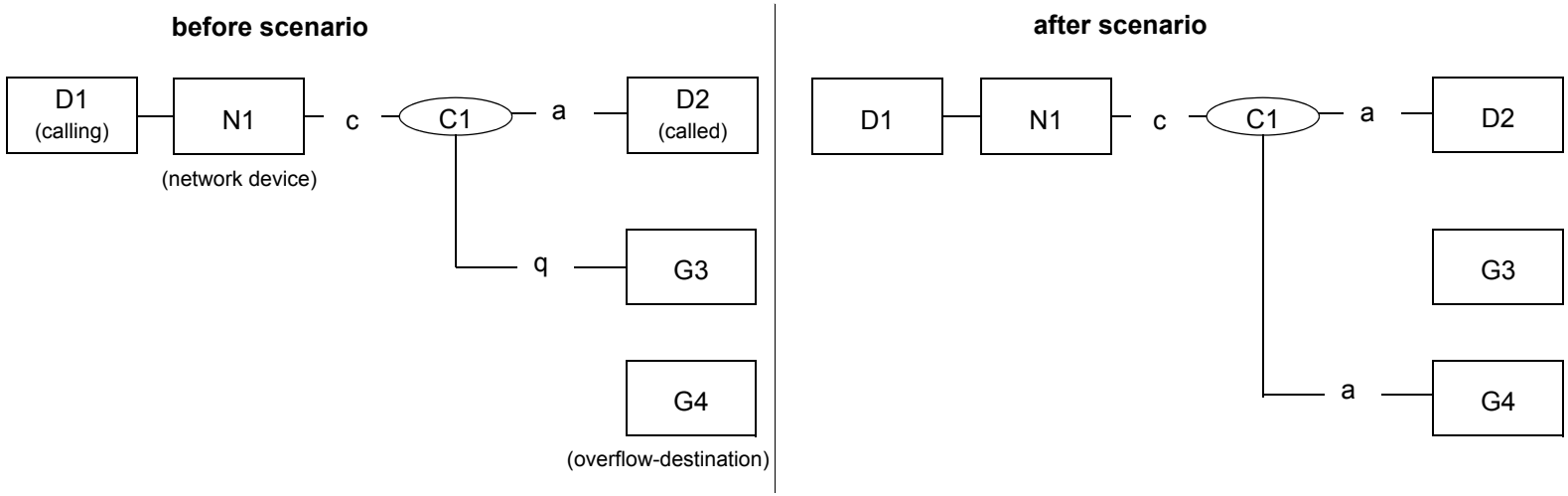
Table 5-76 Intercept with parallel call to GAMQ (page 3 of 3)

Remark:

Immediately after the AC has answered the call, it automatically seizes D2 again in a consultation-call (a new call-id will be created).

5.17.5.8 Intercept with parallel call, call is routed to another GA after timeout

An external caller (directory-number D1) calls D2 via the network-interface N1 (trunk). D2 does not answer the call. After a timeout, the call is intercepted with parallel call to G3 (GA2Q) and queued there. The call is not answered by an AC within a certain amount of time; after a time-out, the call is overflown to GA4 (GAMQ).



Activity	Monitored Device N1	Monitored Device D2	Monitored Device G3 (GA2Q)	Monitored Device G4 (GAMQ)	Comments
1. The call was queued at GA2Q due to intercept with parallel call - an overflow-timer elapses, G3 withdraws from the call.	**Connection Cleared** • droppedConnection G3C1 • releasingDevice G3 • localConnectionInfo connected • cause normalClr • servicesPermitted ClearCall SendUI	**Connection Cleared** • droppedConnection G3C1 • releasingDevice G3 • localConnectionInfo alerting • cause normalClr • servicesPermitted ClearConn Answer SendUI	**Connection Cleared** • droppedConnection G3C1 • releasingDevice G3 • localConnectionInfo null • cause normalClr • servicesPermitted none		**Please note:** no Diverted-Event is sent for G3 because the call remains MultiAlert all the time (G3 leaves the call and D4 joins the call). (Please refer to section 5.17.5.1.7, "General Rules concerning Multi-Alert-Situations for GA" for more information).
2. Overflow-destination G4 is added to the call.	**Delivered** • connection G4C1 • alertingDevice G4 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev D2 • localConnectionInfo connected • cause multiAlert • servicesPermitted ClearCall SendUI	**Delivered** • connection G4C1 • alertingDevice G4 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo alerting • cause multiAlert • servicesPermitted ClearConn Answer SendUI		**Delivered** • connection G4C1 • alertingDevice G4 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo alerting • cause multiAlert • servicesPermitted SendUI	

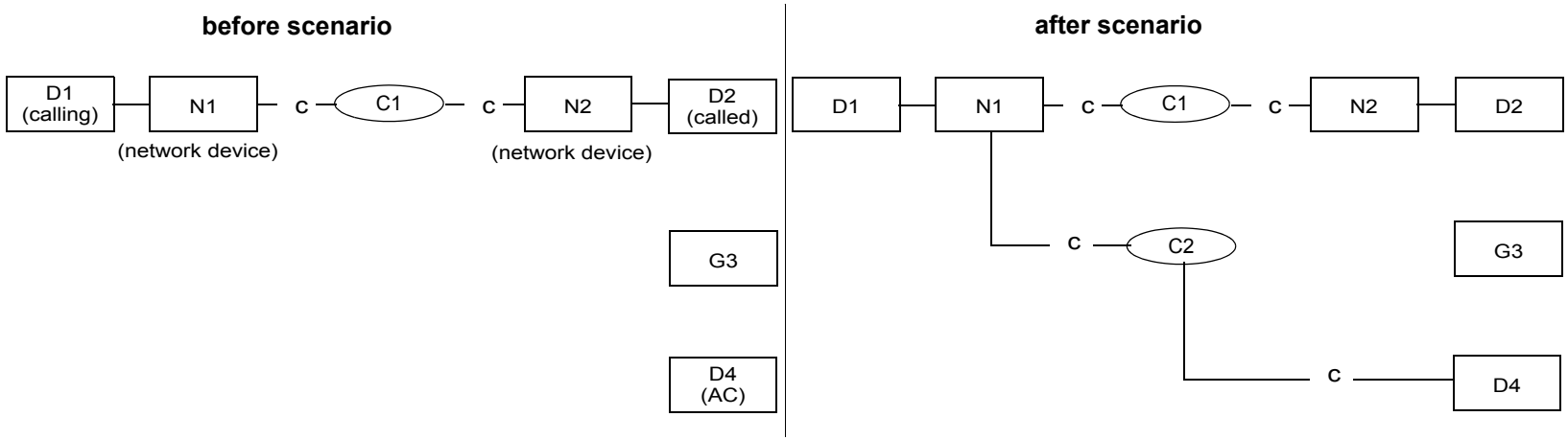
Table 5-77 Intercept with parallel call to GA2Q

Remark:

None

5.17.5.9 Trunk-to-trunk supervision

Two NIDs (trunks) without disconnect supervision are connected to each other (Call-Id C2). When the supervision timer expires for N1, a recall to the GA is executed, an AC answers the call.



Activity	Monitored Device N1 (trunk1)	Monitored Device N2 (trunk2)	Monitored Device G3 (GA)	Monitored Device D4 (AC)	Comments
1. Supervision timer expires for N1(trunk 1)	**No event**	**No event**	**Delivered** • connection G3C2 • alertingDevice G3 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C2 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo alerting • cause enterDist • servicesPermitted SendUI		**No events** are sent for the trunks - this is non-standard-behaviour. (as is the case for all override-scenarios).

Table 5-78 Trunk to trunk supervision (page 1 of 2)

Activity	Monitored Device N1 (trunk1)	Monitored Device N2 (trunk2)	Monitored Device G3 (GA)	Monitored Device D4 (AC)	Comments
2. AC is rung	**No event**	**No event**	**Delivered** • connection D4C2 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C2 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo alerting • cause multiAlert • servicesPermitted SendUI	**Delivered** • connection D4C2 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C2 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev NS • localConnectionInfo alerting • cause multiAlert • servicesPermitted SendUI	
3. D4 answers call - GA withdraws from the call	**No event**	**No event**	**Connection Cleared** • droppedConnection G3C2 • releasingDevice G3 • localConnectionInfo null • cause multiAlert • servicesPermitted none	**Connection Cleared** • droppedConnection G3C2 • releasingDevice G3 • localConnectionInfo alerting • cause multiAlert • servicesPermitted SendUI	
4. D4 is connected to D1	**No event**	**No event**		**Established** • establishedConn D4C2 • answeringDevice D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDev NS • localConnectionInfo connected • cause normal • servicesPermitted SendUI	

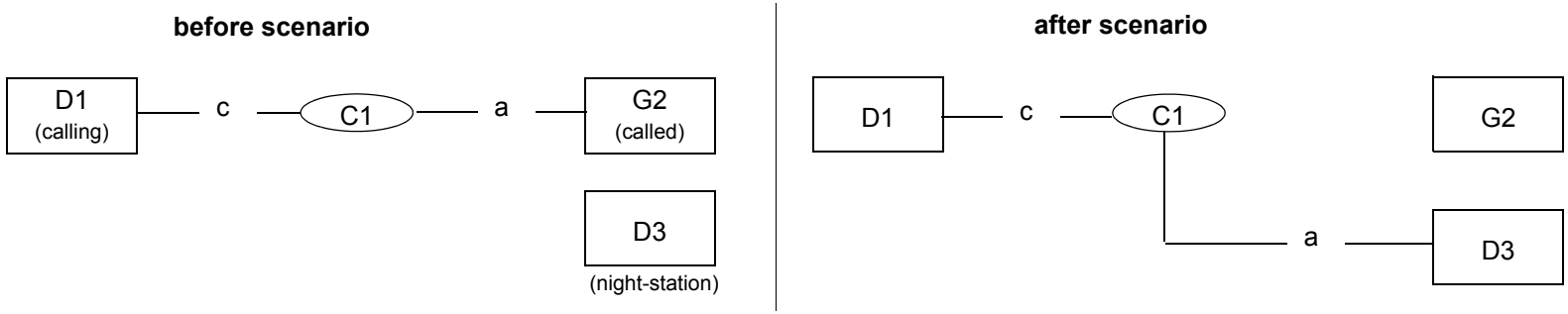
Table 5-78 Trunk to trunk supervision (page 2 of 2)

Remark:

After having answered the call, the AC listens in on the call (like in an override situation) and decides whether to tear down the call or leave it.

5.17.5.10 Night-Service: General Night Station answers

D1 calls G2 (GA2Q, internal attendant access code: G2-int), the GA is in night-mode (General Night Service, GNS). The call is routed to the night-station D3.



Please refer to section 5.17.4.3, “Internal call to HG, Hunt Advance” for the event flow that leads to the „before“-state.

Activity	Monitored Device D1	Monitored Device G2 (GA2Q)	Monitored Device D3	Comments
1. The call was prevously alerting at the GA. GA diverts the call to the night-station		**Diverted** • connection G2C1 • divertingDevice G2 • newDestination D3 • callingDevice D1 • calledDevice G2-int • lastRedirectionDev NS • localConnectionInfo null • cause Cf-NA • servicesPermitted none		**Please note:** A Diverted-Event is sent because the call leaves the GA as soon as the night-station starts ringing.
2. The night station is rung	**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice G2-int • lastRedirectionDev G2 • localConnectionInfo connected • cause Cf-NA • servicesPermitted ClearConn CallBack SendUI		**Delivered** • connection D3C1 • alertingDevice D3 • callingDevice D1 • calledDevice G2-int • lastRedirectionDev G2 • localConnectionInfo alerting • cause Cf-NA • servicesPermitted Answer ClearConn Deflect SendUI	

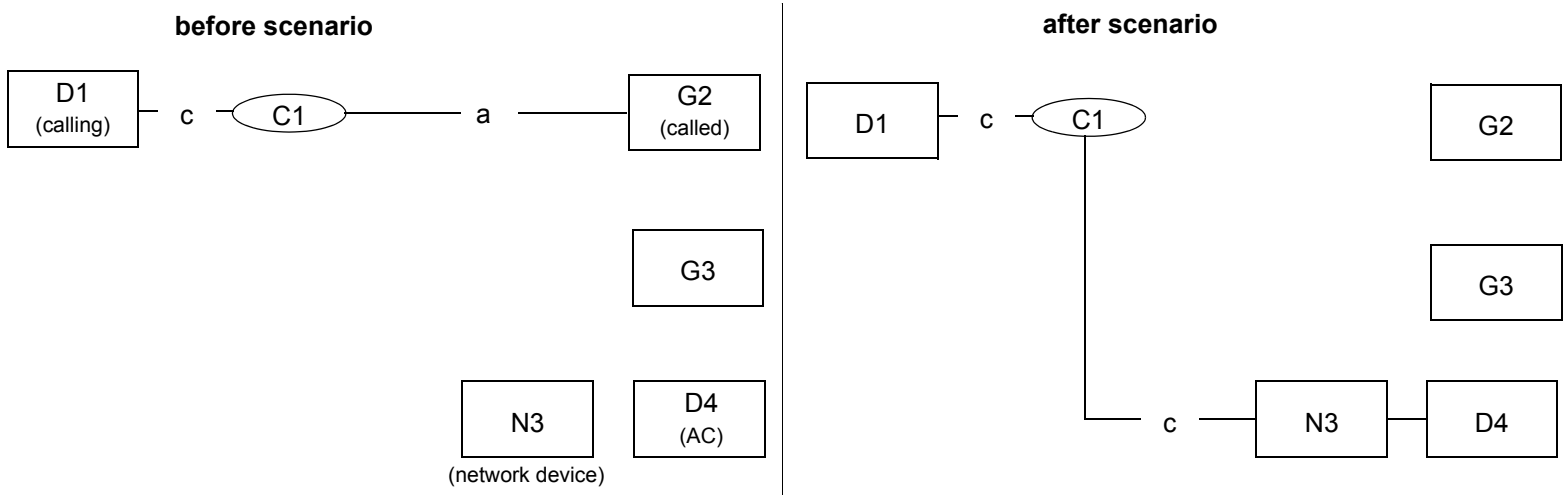
Table 5-79 Night-Service (GNS) - night station answers

Remark:

None

5.17.5.11 Night-Service: ZVFEXT

D1 calls G2 (GA2Q; internal attendant access code, diallable number: G2-int). G2 is in night-service (ZVFEXT). The call is forwarded to a GA2Q on another node (G3) and answered by an AC (D4) on the other node.



Remark:

The call flow for the GA on the other node is like a basic external, incoming call to GA2Q and therefore not reproduced here.

Activity	Monitored Device D1		Monitored Device G2 (GA2Q)		Monitored Device N3 (trunk)		Comments
1. The call leaves the CSTA subdomain.	**NW-Reached**		**NW-Reached**		**NW-Reached**		
	• outboundConn	N3C1	• outboundConn	N3C1	• outboundConn	N3C1	
	• NWInterfaceUsed	N3	• NWInterfaceUsed	N3	• NWInterfaceUsed	N3	
	• callingDevice	D1	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	G2-int	• calledDevice	G2-int	• calledDevice	G2-int	
	• AssCalledDevice	N3	• AssCalledDevice	N3	• AssCalledDevice	N3	
	• lastRedirectionDev	NS	• lastRedirectionDev	NS	• lastRedirectionDev	NS	
	• NW capabilities	ISDN Private	• NW capabilities	ISDN Private	• NW capabilities	ISDN Private	
	• localConnectionInfo	connected	• localConnectionInfo	alerting	• localConnectionInfo	connected	
	• cause	normal	• cause	normal	• cause	normal	
	• servicesPermitted	ClearConn SendUI	• servicesPermitted	SendUI	• servicesPermitted	ClearConn CallBack SendUI	
2. The GA on the other node is alerted.	**Delivered**		**Delivered**		**Delivered**		**Please note:** **G3** (the GA) is alerting on the other node)
	• connection	N3C1	• connection	N3C1	• connection	N3C1	
	• alertingDevice	**G3**	• alertingDevice	**G3**	• alertingDevice	**G3**	
	• callingDevice	D1	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	G2-int	• calledDevice	G2-int	• calledDevice	G2-int	
	• AssCalledDevice	N3	• AssCalledDevice	N3	• AssCalledDevice	N3	
	• lastRedirectionDev	NS	• lastRedirectionDev	NS	• lastRedirectionDev	NS	
	• localConnectionInfo	connected	• localConnectionInfo	alerting	• localConnectionInfo	connected	
	• cause	multiAlert	• cause	multiAlert	• cause	multiAlert	
	• servicesPermitted	ClearConn SendUI	• servicesPermitted	SendUI	• servicesPermitted	ClearConn CallBack SendUI	
3. AC on other node answers the call - the GA leaves the call.	**Connection Cleared**		**Connection Cleared**		**Connection Cleared**		
	• droppedConnection	G2C1	• droppedConnection	G2C1	• droppedConnection	G2C1	
	• releasingDevice	G2	• releasingDevice	G2	• releasingDevice	G2	
	• localConnectionInfo	connected	• localConnectionInfo	null	• localConnectionInfo	connected	
	• cause	multiAlert	• cause	multiAlert	• cause	multiAlert	
	• servicesPermitted	ClearConn Consult, Hold SST GenDg GenTelTon SendUI	• servicesPermitted	none	• servicesPermitted	ClearConn SendUI	

Table 5-80 Night-Service (ZVFEXT) - AC on other node answers (page 1 of 2)

Activity	Monitored Device D1	Monitored Device G2 (GA2Q)	Monitored Device N3 (trunk)	Comments
4. The call is established between D1 and the AC on the other node (via NID N3).	**Established** • establishedConn N3C1 • answeringDevice **D4** • callingDevice D1 • calledDevice G2-int • AssCalledDevice N3 • lastRedirectionDev NS • localConnectionInfo connected • cause NW-signal • servicesPermitted ClearConn Consult, Hold SST GenDg GenTelTon SendUI		**Established** • establishedConn N3C1 • answeringDevice **D4** • callingDevice D1 • calledDevice G2-int • AssCalledDevice N3 • lastRedirectionDev NS • localConnectionInfo connected • cause NW-signal • servicesPermitted ClearConn SendUI	**answering Device = D4:** As soon as the AC on the other node answers, the network-information changes. Therefore, D4 is shown as answering Device.

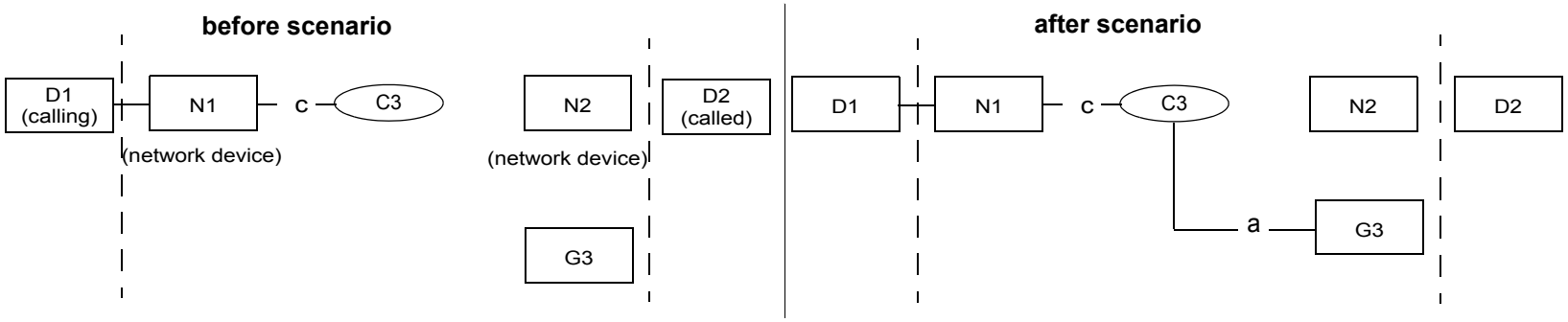
Table 5-80 Night-Service (ZVFEXT) - AC on other node answers (page 2 of 2)

Remark:

None

5.17.5.12 Intercept on a transit node

Node 2 is a transit-node: an external caller (directory-number D1 from node 1) calls a station on node 3(D2). D2 does not answer, the call is intercepted without parallel call to the GA2Q on node 2 G3.



Activity	Monitored Device N1		Monitored Device N2		Monitored Device G3 (GA)	Comments
1. D1 has finished dialling. The call leaves the CSTA subdomain	**NW-Reached** • outboundConn • NWInterfaceUsed • callingDevice • calledDevice • AssCallingDevice • NWCallingDevice • lastRedirectionDev • NW capabilities • localConnectionInfo • cause • servicesPermitted	 N2C1 N2 D1 D2 N1 D1 NS ISDN Private connected normal ClearConn SendUI	**NW-Reached** • outboundConn • NWInterfaceUsed • callingDevice • calledDevice • AssCallingDevice • NWCallingDevice • lastRedirectionDev • NW capabilities • localConnectionInfo • cause • servicesPermitted	 N2C1 N2 D1 D2 N1 D1 NS ISDN Private connected normal ClearConn SendUI		
2. External destination D2 is reached	**Established** • establishedConn • answeringDevice • callingDevice • calledDevice • AssCallingDevice • NWCallingDevice • AssCalledDevice • lastRedirectionDev • localConnectionInfo • cause • servicesPermitted	 N2C1 **D2' (1)** D1 D2 N1 D1 N2 NS connected NW-signal ClearConn SendUI	**Established** • establishedConn • answeringDevice • callingDevice • calledDevice • AssCallingDevice • NWCallingDevice • AssCalledDevice • lastRedirectionDev • localConnectionInfo • cause • servicesPermitted	 N2C1 **D2' (1)** D1 D2 N1 D1 N2 NS connected NW-signal ClearConn SendUI		**Established**: Whenever HiPath 4000 connects two trunks, this results in Established-Events for both trunks, even if the call was not answered yet **(1) D2':** The answeringDev might differ from the calledDevice, even if no forwarding or similar actions were performed on node 3. This is due to information ACL receives from the network.

Table 5-81 Intercept on a transit node (page 1 of 2)

Activity	Monitored Device N1	Monitored Device N2	Monitored Device G3 (GA)	Comments
3. Intercept-timer expires, the call is intercepted to GA		**Diverted** • connection N2C1 • divertingDevice D2' • newDestination G3 • callingDevice D1 • calledDevice D2 • lastRedirectionDev NS • localConnectionInfo null • cause callNotAnswer • servicesPermitted none		
4. The GA is alerted.	**Delivered** • connection G3C1 • alertingDevice G3 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev D2 • localConnectionInfo connected • cause enterDist • servicesPermitted ClearCall SendUI		**Delivered** • connection G3C1 • alertingDevice G3 • callingDevice D1 • calledDevice D2 • OrigNIDConn N1C1 • NWCallingDevice D1 • AssCallingDevice N1 • lastRedirectionDev D2 • localConnectionInfo alerting • cause enterDist • servicesPermitted SendUI	

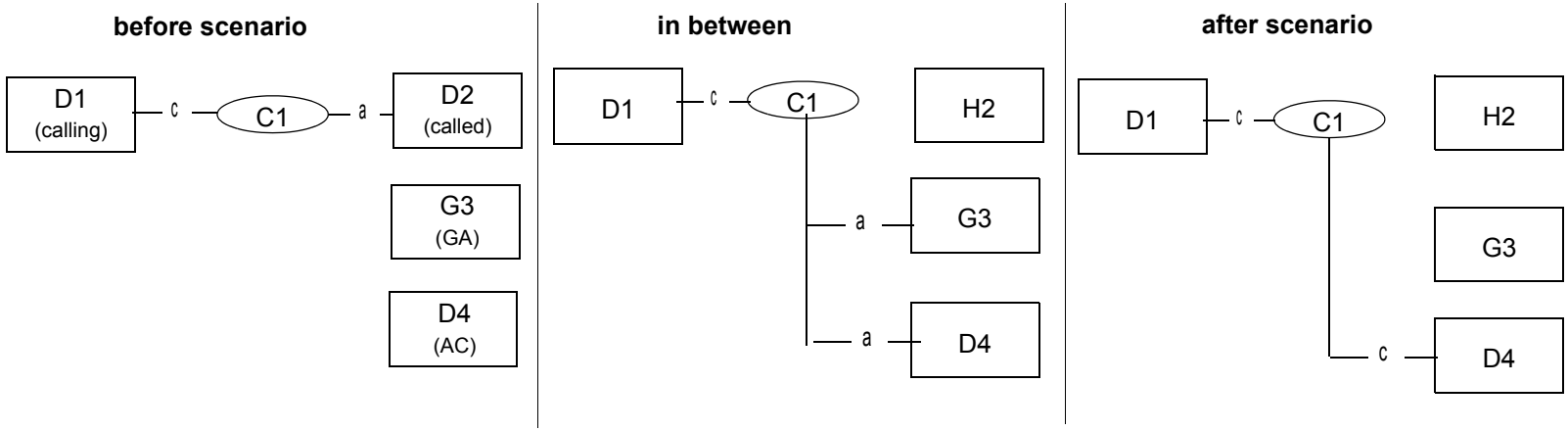
Table 5-81 Intercept on a transit node (page 2 of 2)

Remark:

None

5.17.5.13 Call forwarded (CFNA) to Attendant, Multiple alerting, providing Diverted event to calling side is enabled

D1 calls D2 which has call forward no answer configured to GA (G3), call is answered by an AC (D4).



Activity	Monitored Device D1	Monitored Device D2	Monitored Device G3 (General Attendant	Monitored Device D4	Comments
1 CFNA timer expires	**DivertedEvent** • connection D2C1 • divertingDevice D2 • newDest G3 • callingDevice D1 • calledDevice D2 • lastRedirectionDev NS • localConnectionInfo connected • cause forwardNoAns • servicesPermitted none	**DivertedEvent** • connection D2C1 • divertingDevice D2 • newDest G3 • callingDevice D1 • calledDevice D2 • lastRedirectionDev NS • localConnectionInfo null • cause forwardNoAns • servicesPermitted none			
2. General attendant is in alerting state	**DeliveredEvent** • connection G3C1 • alertingDevice G3 • callingDevice D1 • calledDevice D2 • lastRedirectionDev D2 • localConnectionInfo connected • cause enterDist • servicesPermitted ClearConn SendUI		**DeliveredEvent** • connection G3C1 • alertingDevice G3 • callingDevice D1 • calledDevice D2 • lastRedirectionDev D2 • localConnectionInfo alerting • cause enteDist • servicesPermitted SendUI		Please note: No Deflect is allowed for D3 because this is a MultiAlert-sitation

Activity	Monitored Device D1	Monitored Device D2	Monitored Device G3 (General Attendant	Monitored Device D4	Comments
3. Attendant console starts alerting	**DeliveredEvent** • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDev D2 • localConnectionInfo connected • cause multiAlert • servicesPermitted ClearConn SendUI		**DeliveredEvent** • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDev D2 • localConnectionInfo alerting • cause multiAlert • servicesPermitted SendUI	**DeliveredEvent** • connection D4C1 • alertingDevice D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDev D2 • localConnectionInfo alerting • cause multiAlert • servicesPermitted SendUI	Please note: No Deflect is allowed for D3 because this is a MultiAlert-sitation
4. D4 answers call - GA-device leaves the call	**Connection Cleared** • droppedConnection G3C1 • releasingDevice G3 • localConnectionInfo connected • cause multiAlert • servicesPermitted ClearConn Consult, Hold SST GenDg GenTelTon SendUI	• • • • •	**Connection Cleared** • droppedConnection H2C1 • releasingDevice H2 • localConnectionInfo null • cause multiAlert • servicesPermitted none	**Connection Cleared** • droppedConnection G3C1 • releasingDevice G3 • localConnectionInfo alerting • cause multiAlert • servicesPermitted ClearConn Consult, Hold SST GenDg GenTelTon SendUI	
5. D4 is connected to the original call	**Established** • establishedConn D4C1 • answeringDevice D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDev NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn Consult, Hold SST GenDg GenTelTon SendUI			**Established** • establishedConn D4C1 • answeringDevice D4 • callingDevice D1 • calledDevice D2 • lastRedirectionDev NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn Consult Hold SST GenDg GenTelTon SendUI	

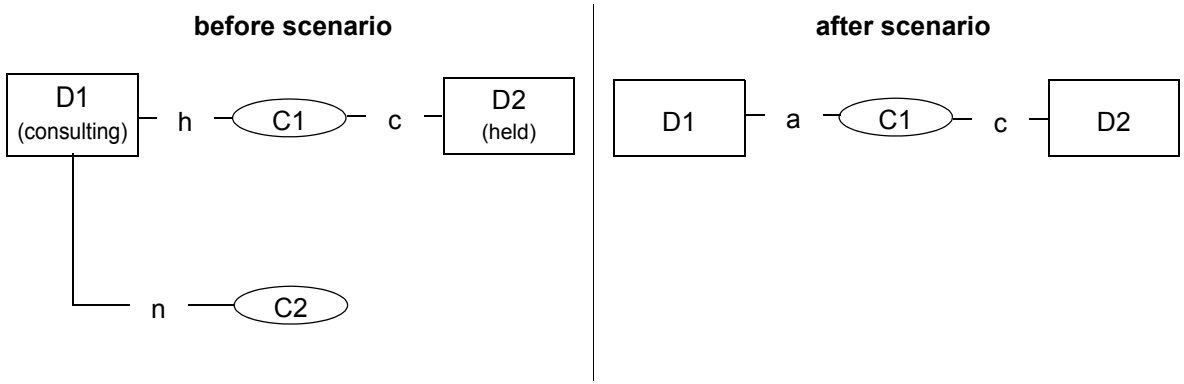
Remark:

None

5.18 Recall Scenarios

5.18.1 Softhold Recall

The consulting party clears its secondary call and it will be immediately recalled by the held party.



Activity	Monitored Device D1		Monitored Device D2		Comments
1. D1 will be recalled.	**Delivered**		**Delivered**		
	• deliveredConnection	D1C1	• deliveredConnection	D1C1	
	• alertingDevice	D1	• alertingDevice	D1	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	alerting	• localConnectionInfo	connected	
	• cause	recall	• cause	recall	
	• servicesPermitted	Answer, ClearConn, Deflect,SendUserInfo	• servicesPermitted	CallBack, ClearConn, SendUserInfo	

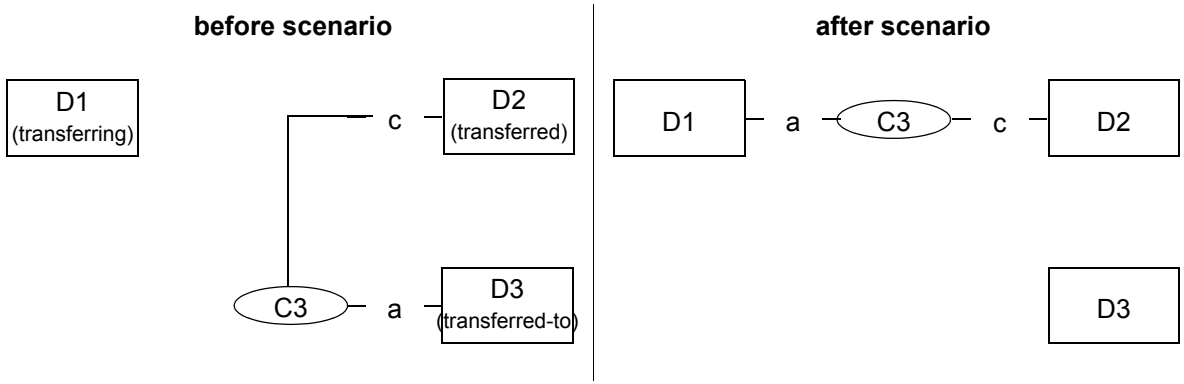
Table 5-82 Softhold Recall

Remark:

None

5.18.2 Transfer Recall

If the transferred-to party does not answer until a specified time interval, the transferring device will be recalled by the transferred party.



See “Blind Transfer (with local view in Transferred event)” on page 5-84 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Since device D3 does not answer, device D1 will be recalled.			**Diverted** • divertedConnection D3C3 • divertingDevice D3 • newDestinationDevice D1 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo null • cause recall • servicesPermitted none	The switching function sends the Diverted event only to the divertingDevice.

Table 5-83 Transfer Recall (page 1 of 2)

Activity	Monitored Device D1		Monitored Device D2		Monitored Device D3	Comments
2. D1 alerts.	**Delivered**		**Delivered**			
	• deliveredConnection	D1C3	• deliveredConnection	D1C3		
	• alertingDevice	D1	• alertingDevice	D1		
	• callingDevice	D2	• callingDevice	D2		
	• calledDevice	D3	• calledDevice	D3		
	• lastRedirectionDevice	D3	• lastRedirectionDevice	D3		
	• localConnectionInfo	alerting	• localConnectionInfo	connected		
	• cause	recall	• cause	recall		
	• servicesPermitted	Answer, ClearConn, Deflect, DialDgt, SendUserInfo	• servicesPermitted	CallBack, ClearConn, SendUserInfo		

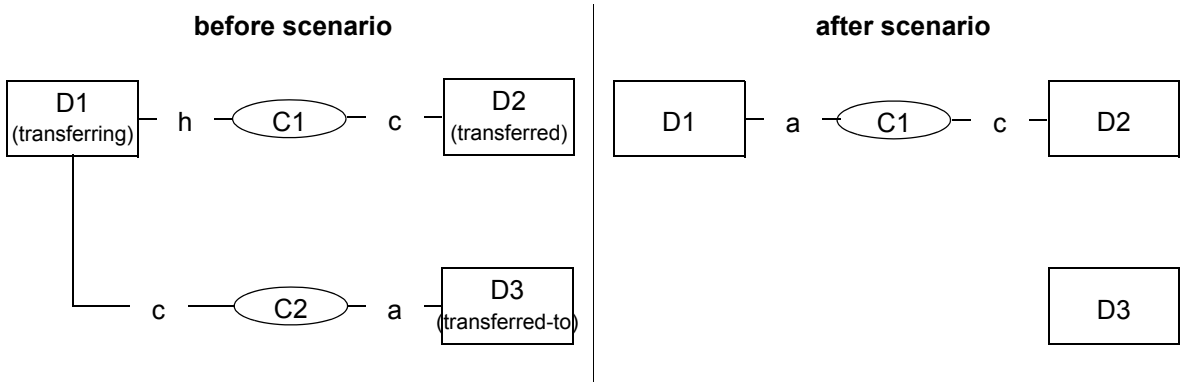
Table 5-83 Transfer Recall (page 2 of 2)

Remark:

None

5.18.3 Transfer with Restricted Connection

As the connection between the transferred and the transferred-to device is restricted, the transfer will result in a recall from the transferred device to the transferring device.



See “Successful consultation Call” on page 5-68 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. D1 goes on hook to transfer, but it is not possible, so the call will be cleared.	**Connection Cleared** • droppedConnection D1C2 • releasingDevice D1 • localConnectionInfo null • cause callNotAnswered • servicesPermitted ClearConn		**Connection Cleared** • droppedConnection D1C2 • releasingDevice D1 • localConnectionInfo alerting • cause callNotAnswered • servicesPermitted none	
2. Device D3 is cleared from the call.			**Connection Cleared** • droppedConnection D3C2 • releasingDevice D3 • localConnectionInfo null • cause normalClr • servicesPermitted none	
3. D2 will be recalled.	**Delivered** • deliveredConnection D1C1 • alertingDevice D1 • callingDevice D2 • calledDevice D1 • lastRedirectionDevice NS • localConnectionInfo alerting • cause recall • servicesPermitted Answer, ClearConn, Deflect, DialDgt, SendUserInfo	**Delivered** • deliveredConnection D1C1 • alertingDevice D1 • callingDevice D2 • calledDevice D1 • lastRedirectionDevice NS • localConnectionInfo connected • cause recall • servicesPermitted CallBack, ClearConn, SendUserInfo		

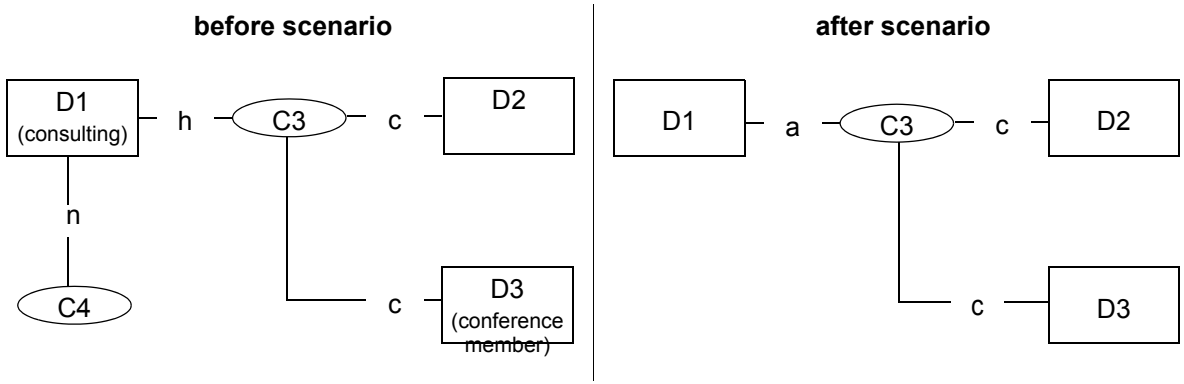
Table 5-84 Transfer with Restricted Connection

Remark:

None

5.18.4 Conference Recall

A participating party of the conference consults and afterwards clears its secondary call. It will be immediately recalled by the conference.



Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Device D1 is recalled.	**Delivered** • deliveredConnection D1C3 • alertingDevice D1 • callingDevice NK • **calledDevice D2** • lastRedirectionDevice NS • localConnectionInfo alerting • cause recall • servicesPermitted ClearConn, Answer, Deflect, DialDgt, SendUserInfo	**None**	**None**	Note that only the recalled party gets the Delivered event. If D1 answers, the Established event will also be reported only for D1. Note, that the originally called device remains D2.

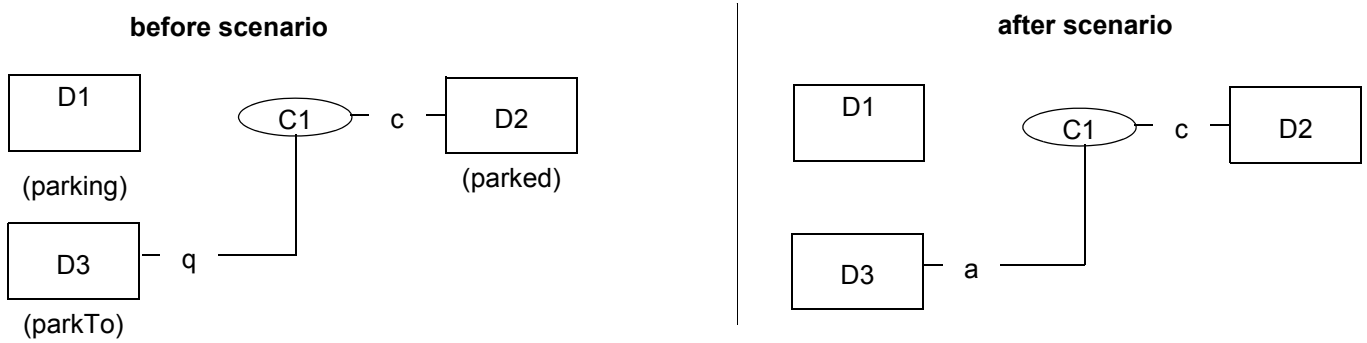
Table 5-85 Conference Recall

Remark:

None

5.18.5 Park Timer expires

After the park timer expires, the parkTo party will be notified of the parked party.



See “Manual directed park call” on page 5-59 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2		Monitored Device D3		Comments
1. Device D3 starts ringing after the park timer expired.	**none**	**Delivered**		**Delivered**		
		• connection	D3C1	• connection	D3C1	
		• alertingDevice	D3	• alertingDevice	D3	
		• callingDevice	D1	• callingDevice	D1	
		• calledDevice	D2	• calledDevice	D2	
		• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
		• localConnectionInfo	connected	• localConnectionInfo	alert	
		• cause	normal	• cause	normal	
		• servicesPermitted	CallBack, ClearConn, SendUserInfo	• servicesPermitted	Answer, ClearConn, Deflect, SendUserInfo	

Table 5-86 Park timer expires

Remark:

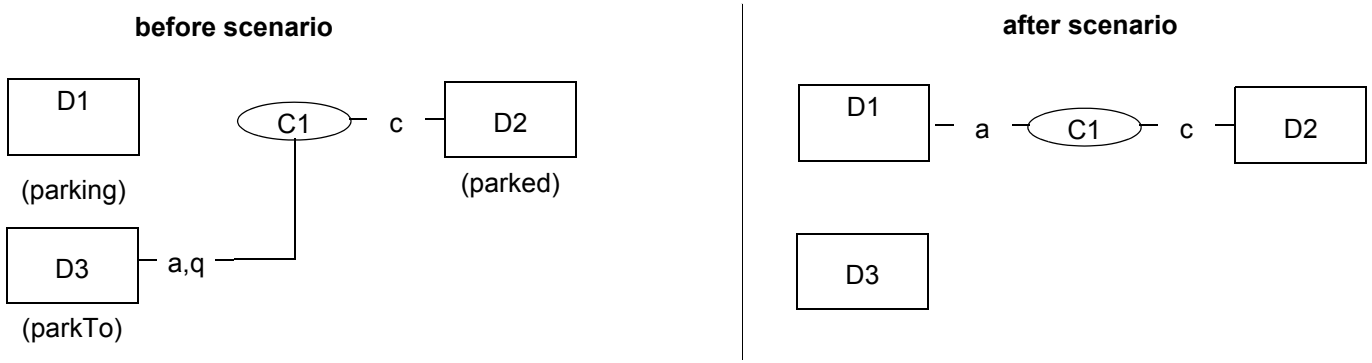
The switching function does not change the calling, called parameters in the event flow.

ECMA TR/82 reports changing calling, called devices in the related scenario.

See “Manual directed park call” on page 5-59.

5.18.6 Park Recall Timer Expires

After the park recall timer expires, the parked party recalls the parking party .



See “Park Timer expires” on page 5-188 or “Manual directed park call” on page 5-59 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. The call has been diverted from D3 due to the park recall timer expired.			**Diverted** • connection D3C1 • divertingDevice D3 • newDestination D1 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NS • localConnectionInfo null • cause recall • servicesPermitted none	The switching function sends the Diverted event only to the divertingDevice.

Table 5-87 Park Recall Timer Expires (page 1 of 2)

Activity	Monitored Device D1		Monitored Device D2		Monitored Device D3	Comments
2. D1 alerts.	**Delivered**		**Delivered**			
	• connection	D1C1	• connection	D1C1		
	• alertingDevice	D1	• alertingDevice	D1		
	• callingDevice	D1	• callingDevice	D1		
	• calledDevice	D2	• calledDevice	D2		
	• lastRedirectionDevice	D3	• lastRedirectionDevice	D3		
	• localConnectionInfo	alert	• localConnectionInfo	connected		
	• cause	recall	• cause	recall		
	• servicesPermitted	Answer, ClearConn, Deflect, SendUserInfo	• servicesPermitted	CallBack, ClearConn, SendUserInfo		

Table 5-87 Park Recall Timer Expires (page 2 of 2)

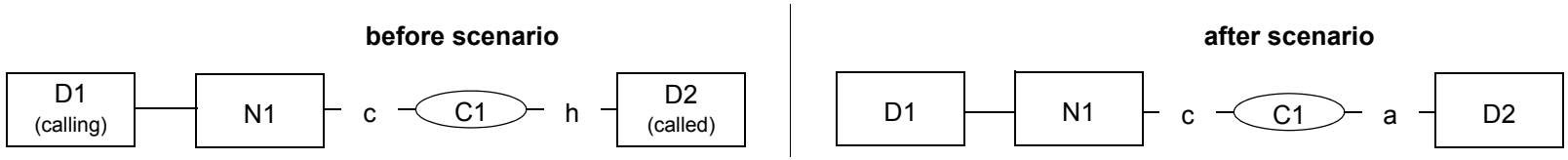
Remark:

None

5.18.7 Hard Hold Recall

This scenario describes the event flow of the Hard Hold Recall feature. An analog device puts an incoming external call on Hard Hold. After the timer expires the external party recalls the analog device.

D1 is anate (analog telephone).



Activity	Monitored Device N1		Monitored Device D2		Comments
1. Network device N1 recalls device D2.	**Delivered**		**Delivered**		
	• connection	D2C1	• connection	D2C1	
	• alertingDevice	D2	• alertingDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• origNID connID	N1C1	• origNID connID	N1C1	
	• localConnectionInfo	connected	• localConnectionInfo	alerting	
	• cause	recall	• cause	recall	
	• servicesPermitted	CallBack, ClearConn, SendUserInfo	• servicesPermitted	Answer, ClearConn, Deflect, DialDgt, SendUserInfo	

Table 5-88 Hard Hold Recall

Remark:

None

5.19 OpenScape Specific Features

This section describes OpenScape 4000 features, that are either not described by ECMA 269 or they are not CSTA III standard compliant.

5.19.1 Route Optimization

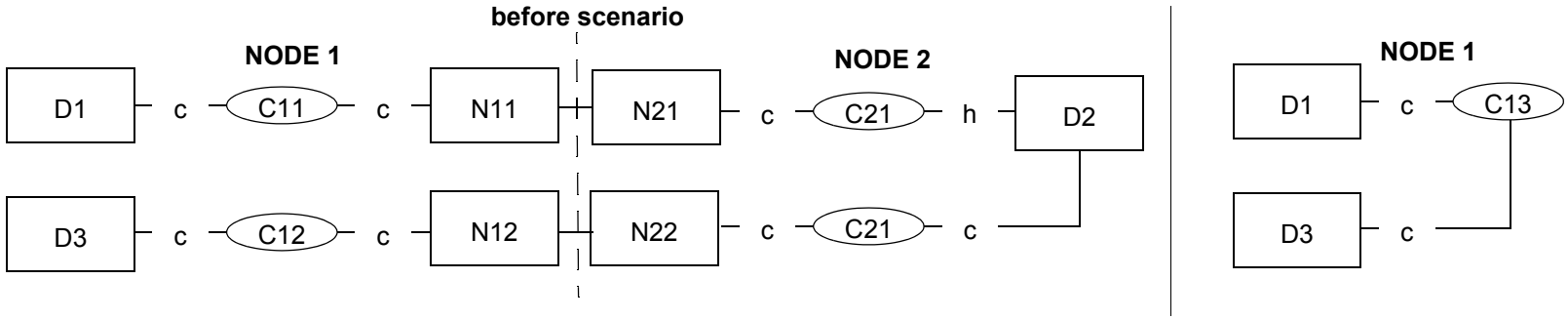
This scenario describes an event flow, where the local route optimization feature is performed.

Device D1 and D2 are connected via network interface device N11. Device D2 consults to D3 and transfers D1 to D3. The switch optimizes the route to the local node and releases the involved network interface devices.

The Route Optimization feature is executed in the following cases:

1. After a netwide Transfer
2. After a netwide Pickup
3. After a netwide Park

Note , that in case of an ACD call Route Optimization is not possible.



Activity	Monitored Device D1	Monitored Device N11	Monitored Device D2	Monitored Device N12	Comments
1. D2 hits transfer. The route will be optimized.	**Transferred** • primaryOldCall D1C11 • transferring N11 • transferredTo D3 • transferredConn 1. new / old (D1C13) / (D1C11) 2. new (D3C13) • localConnection connected • cause SST • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	**Transferred** • primaryOldCall N11C11 • transferring N11 • transferredTo NK • transferredConn 1.old N11C11 • localConnection null • cause SST • servicesPermitted none	**Transferred** • primaryOldCall D3C12 • transferring N12 • transferredTo D3 • transferredConn 1. new / old (D3C13) / (D3C12) 2. new (D1C13) • localConnection connected • cause SST • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	**Transferred** • primaryOldCall N12C12 • transferring N12 • transferredTo NK • transferredConn 1. old N12C12 • localConnection null • cause SST • servicesPermitted none	The event flow of the route optimization is not compliant with the ECMA 269 standard.
2. Device D1 and D3 are connected locally in a call.	**Established** • establishedConn D3C13 • answeringDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirection NS • localConnectionInfo connected • cause NWSignal • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo		**Established** • establishedConn D3C13 • answeringDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirection NS • localConnectionInfo connected • cause NWSignal • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo		

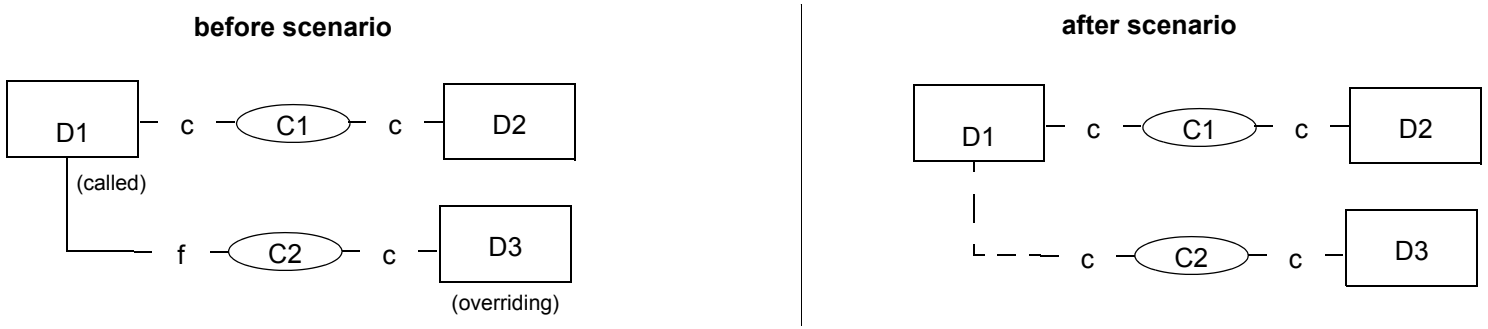
Table 5-89 Route Optimization

Remark:

None

5.19.2 Override

The D3 overriding party intrudes in an established call (C1) by pressing the override key. The called party will have 2 active calls.



Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. The busy connection is cleared immediately.	**Connection Cleared** • droppedConnection D1C2 • releasingDevice D2 • localConnectionInfo null • cause normalClr • servicesPermitted ClearConn	**None**	**Connection Cleared** • droppedConnection D1C2 • releasingDevice D2 • localConnectionInfo connected • cause normalClr • servicesPermitted none	
1. D3 hits override.			**Established** • establishedConnection D1C2 • answeringDevice D1 • callingDevice D3 • calledDevice D1 • lastRedirectionDevice NS • localConnectionInfo connected • cause override • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	The Established event is only reported on the overrinding device monitor.

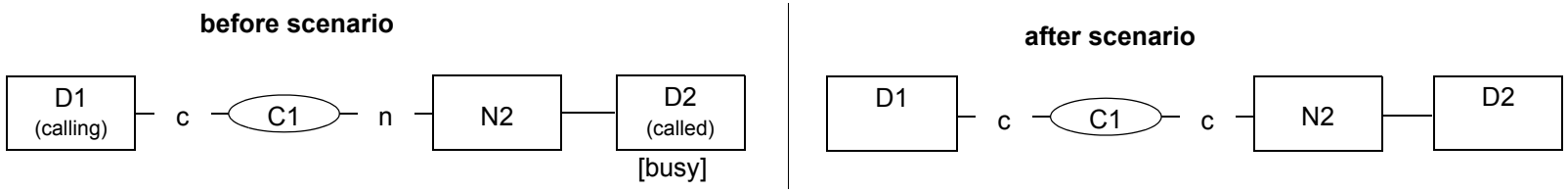
Table 5-90 Override

Remark:

None

5.19.2.1 Netwide override

This scenario illustrates on override to a network party.



See “Manual call to a device outside the CSTA subdomain” on page 5-24 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device N2		Comments
1. The call reaches the network again.		**Network Reached**		The Network Reached event is not provided by the switching function on the monitor of device D1 .
		• outboundConnection	N2C1	
		• networkInterfaceUsed	N2	
		• callingDevice	D1	
		• calledDevice	D2	
		• lastRedirectionDevice	NS	
		• localConnectionInfo	connected	
		• cause	normal	
		• servicesPermitted	ClearConn, SendUserInfo	

Table 5-91 Netwide override (page 1 of 2)

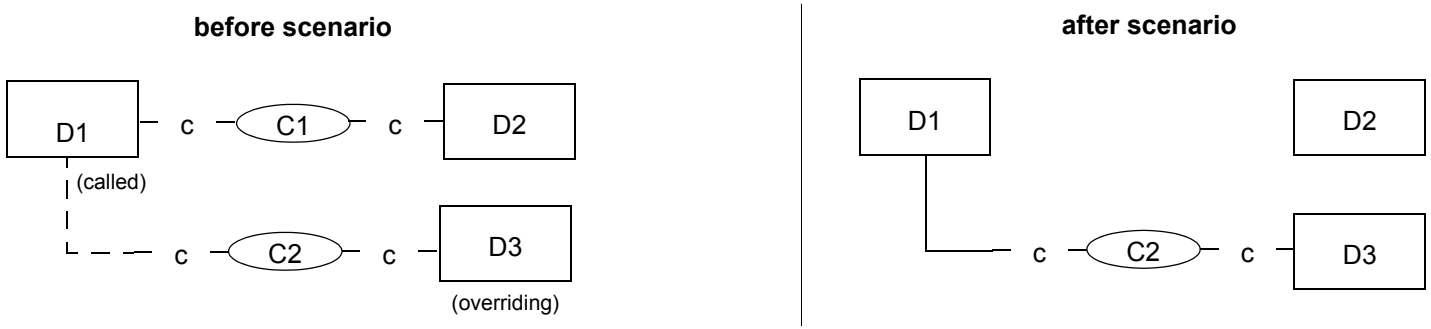
Activity	Monitored Device D1		Monitored Device N2		Comments
2. The connection will be estabilished immediatelly.	**Established**		**Established**		
	• establishedConnection	N2C1	• establishedConnection	N2C1	
	• answeringDevice	D2	• answeringDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	connected	
	• cause	nwSignal	• cause	nwSignal	
	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDgt, GenTelTone, SendUserInfo	• servicesPermitted	ClearConn, SendUserInfo	
	• assocCalledDevice	N2	• assocCalledDevice	N2	

Table 5-91 Netwide override (page 2 of 2)

Remark:

Due to switching function limitation the Network Reached event cannot be provided for both participants.

5.19.2.2 D2 goes onhook after the override



See “Override” on page 5-195 for the event flow to get into the “before scenario” state.

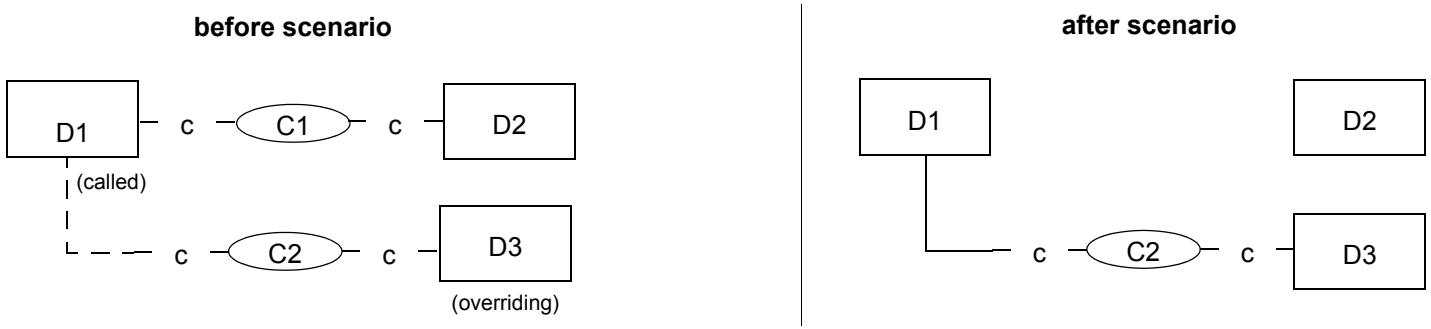
Activity	Monitored Device D1		Monitored Device D2		Monitored Device D3	Comments
1. Device D2 goes onhook.	**Connection Cleared** • droppedConnection • releasingDevice • localConnectionInfo • cause • servicesPermitted	 D2C1 D2 null normalClr ClearConn	**Connection Cleared** • droppedConnection • releasingDevice • localConnectionInfo • cause • servicesPermitted	 D2C1 D2 null normalClr none	**None**	The Connection Cleared for D1C1 is missing.
2. The override tone is stopped,D1 is connected immidiately to D3.	**Established** • establishedConnection • answeringDevice • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted	 D1C2 D1 D3 D1 NS connected normal ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo				The Established is reported only on the called device monitor.

Table 5-92 D2 goes on-hook after the override

Remark:

None

5.19.2.3 D1 hits clear after the override



See “Override” on page 5-195 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Device D1 hits the clear key.	**Connection Cleared** • droppedConnection D1C1 • releasingDevice D1 • localConnectionInfo null • cause normalClr • servicesPermitted ClearConn	**Connection Cleared** • droppedConnection D1C1 • releasingDevice D1 • localConnectionInfo connected • cause normalClr • servicesPermitted ClearConn		
2. As a result of D1C1 clearing, remaining device D2 goes blocked.		**Failed** • failedConnection D2C1 • failingDevice D2 • callingDevice D2 • calledDevice D1 • lastRedirectionDevice NS • localConnectionInfo fail • cause blocked • servicesPermitted ClearConn		
3. As a result of D1C1 clearing, D2C1 is also cleared.		**Connection Cleared** • droppedConnection D2C1 • releasingDevice D2 • localConnectionInfo null • cause normalClr • servicesPermitted none		
4. The override tone is stopped,D1 is connected immidiately to D3.	**Established** • establishedConnection D1C2 • answeringDevice D1 • callingDevice D3 • calledDevice D1 • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo		**Established** • establishedConnection D1C2 • answeringDevice D1 • callingDevice D3 • calledDevice D1 • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	

Table 5-93 D1 hits clear after the override

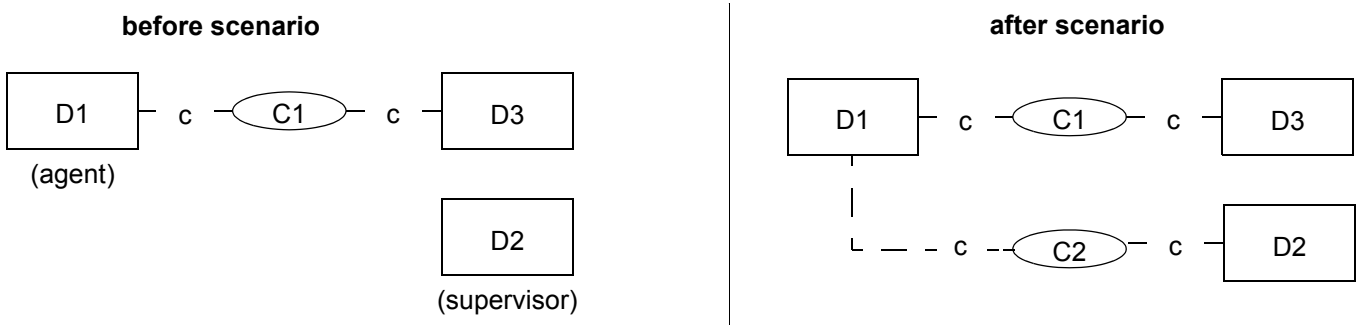
Remark:

None

5.19.3 Silent Monitor

This scenario describes an event flow where silent monitoring is used.

An ACD agent has a call. The supervisor hits the silent monitor key and dials the number of the agent. The supervisor will be connected in the call, and it can listen into the conversation of the agent.



Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. D2 supervisor presses the silent monitor key and dials the agent.(1234)	**None**	**Service Initiated** • initiatedConnection D2C2 • initiatingDevice D2 • localConnectionInfo initiated • cause consultation • servicesPermitted ClearConn, DialDgt	**None**	
		Digits Dialled • diallingConnection D2C2 • diallingDevice D2 • diallingSequence “1234“ • localConnectionInfo initiated • cause normal • servicesPermitted none		

Table 5-94 Silent Monitoring (page 1 of 2)

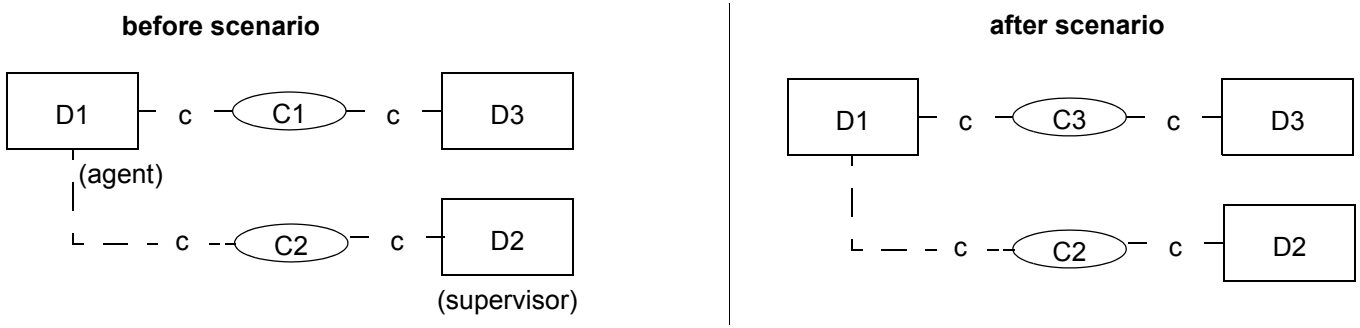
Activity	Monitored Device D1	Monitored Device D2		Monitored Device D3	Comments
2. Device D2 is connecting to the agent.		**Held**			
		• heldConnection	D1C2		
		• holdingDevice	D1		
		• localConnectionInfo	connected		
		• cause	silentParticipation		
		• servicesPermitted	ClearConn		
3. Device D2 is connected to the agent.		**Retrieved**			
		• retrievedConnection	D1C2		
		• Retrieving	D1		
		• localConnectionInfo	connected		
		• cause	silentParticipation		
		• servicesPermitted	ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo		

Table 5-94 Silent Monitoring (page 2 of 2)

Remark:

None

5.19.3.1 The agent goes onhook and afterwards the original caller calls the agent again.



Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Device D1, the agent goes on-hook.	**Connection Cleared** • droppedConnection D1C1 • releasingDevice D1 • localConnectionInfo null • cause normalClr • servicesPermitted ClearConn		**Connection Cleared** • droppedConnection D1C1 • releasingDevice D1 • localConnectionInfo connected • cause normalClr • servicesPermitted ClearConn	
2. The Held event shows, that the agent is idle.		**Held** • heldConnection D1C2 • holdingDevice D1 • localConnectionInfo connected • cause silentParticipation • servicesPermitted ClearConn		
3. As a result of D1C1 clearing, remaining device D3 goes blocked.			**Failed** • failedConnection D3C1 • failingDevice D3 • callingDevice D3 • calledDevice RCG intDNIS • lastRedirectionDevice NS • localConnectionInfo fail • cause blocked • servicesPermitted ClearConn	
4. As a result of D1C1 clearing, D3C1 is also cleared.			**Connection Cleared** • droppedConnection D3C1 • releasingDevice D3 • localConnectionInfo null • cause normalClr • servicesPermitted none	
5. A new call arrives at the RCG which is distributed to the agent.				

Table 5-95 The agent goes onhook and afterwards the original caller calls the agent again. (page 1 of 2)

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
6. D1 starts ringing.	**Delivered** • deliveredConnection D1C3 • alertingDevice D1 • callingDevice D3 • calledDevice RCG intDNIS • lastRedirectionDevice NS • localConnectionInfo alert • cause distributed • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo, DialDgt		**Delivered** • deliveredConnection D1C3 • alertingDevice D1 • callingDevice D3 • calledDevice RCG intDNIS • lastRedirectionDevice NS • localConnectionInfo connected • cause distributed • servicesPermitted CallBack, ClearConn, SendUserInfo	
7. D1 answers the call.	**Established** • establishedConnection D1C3 • answeringDevice D1 • callingDevice D3 • calledDevice RCG intDNIS • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo		**Established** • establishedConnection D1C3 • answeringDevice D1 • callingDevice D3 • calledDevice RCG intDNIS • lastRedirectionDevice NS • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	
8. The Retrieved event shows that the agent has a call.		**Retrieved** • retrievedConnection D1C2 • Retrieving D1 • localConnectionInfo connected • cause normal • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo		

Table 5-95 The agent goes onhook and afterwards the original caller calls the agent again. (page 2 of 2)

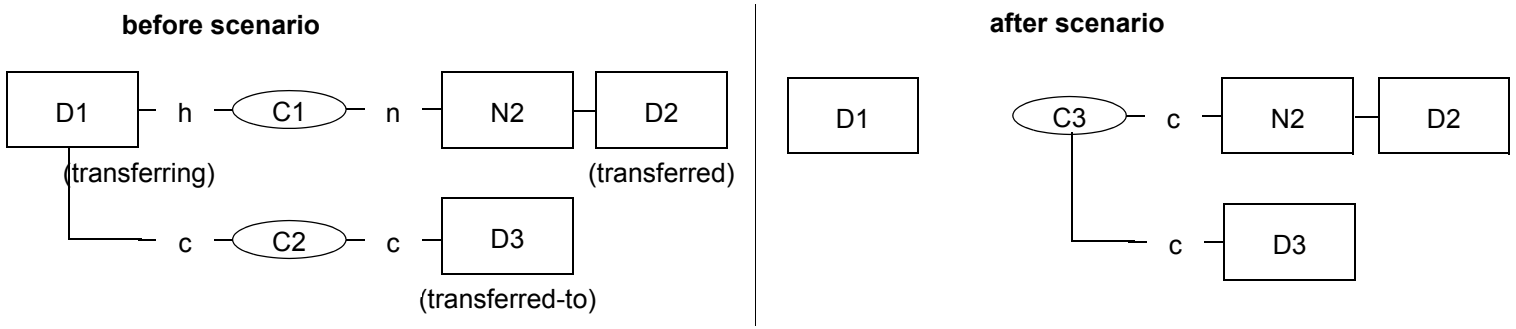
Remark:

None

5.19.4 Transfer Before ALERT

This scenario describes an event flow where the transfer before alert feature is performed.

Device D1 calls D2 via network device N2. In the rare case of very slow network interfaces and appropriate OpenScape 4000 configuration, device D1 can consult to D3 before N2 connects to the call (C1) . Then D1 makes a screened transfer.



Activity	Monitored Device D1		Monitored Device N2	Monitored Device D3		Comments
1. D1 transfers by going on-hook.	**Transferred**			**Transferred**		The transferredConnections parameter only includes the transferredToConnectionID.
	• primaryOldCall	D1C1		• primaryOldCall	D3C2	
	• secondaryOldCall	D1C2				
	• transferringDevice	D1		• transferringDevice	D1	
	• transferredToDevice	D3		• transferredToDevice	D3	
	• transferredConnections 1. new / old	(D3C3) / (D3C2)		• transferredConnections 1. new / old	(D3C3) / (D3C2)	
	• localConnectionInfo	null		• localConnectionInfo	connected	
	• cause	Transfer		• cause	Transfer	
	• servicesPermitted	none		• servicesPermitted	ClearConn	

Table 5-96 Transfer before answer(page 1 of 2)

Activity	Monitored Device D1	Monitored Device N2		Monitored Device D3		Comments
2. The call reaches the network.		**Network Reached**		**Network Reached**		
		• outboundConnection	N2C3	• outbound connection	N2C3	
		• NID device	N2	• NID device	N2	
		• callingDevice	D3	• callingDevice	D3	
		• calledDevice	D2	• calledDevice	D2	
		• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
		• localConnectionInfo	connected	• localConnectionInfo	connected	
		• cause	normal	• cause	normal	
		• servicesPermitted	ClearConn	• servicesPermitted	ClearConn	
3. D2 starts ringing.		**Delivered**		**Delivered**		
		• deliveredConnection	N2C3	• deliveredConnection	N2C3	
		• alertingDevice	D2	• alertingDevice	D2	
		• callingDevice	D3	• callingDevice	D3	
		• calledDevice	D2	• calledDevice	D2	
		• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
		• localConnectionInfo	connected	• localConnectionInfo	connected	
		• cause	NWSignal	• cause	NWSignal	
		• assocCalled	N2	• assocCalled	N2	
		• servicesPermitted	ClearConn, SendUserInfo	• servicesPermitted	CallBack, ClearConn, SendUserInfo	

Table 5-96 Transfer before answer(page 2 of 2)

Remark:

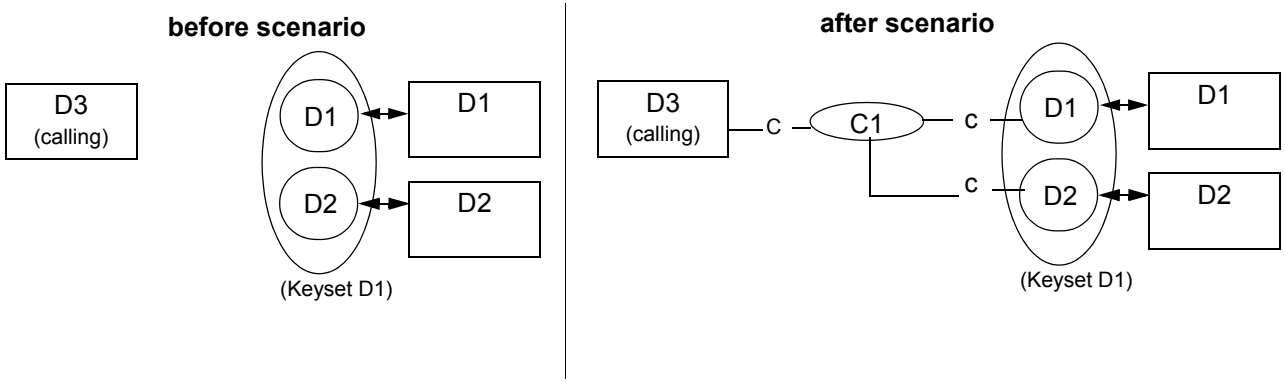
None

5.19.5 Keyset Call (Multiline device call)

Logical device D1 (keyset) has two appearances associated with devices D1 and D2. D1 is the so called primary line, D2 is the secondary line of the keyset.

D3 dials keyset D1. D1/D1 answers the call. D1/D2 joins the call.

OpenScape CA 4000 V7.0 cannot handle multiple appearances, so the below event flow is non standard.



Activity	Monitored Device D3		Monitored Device D1	Comments
1. D3 goes off-hook.	**Service Initiated**			
	• initiatedConnection	D3C1		
	• initiatingDevice	D3		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	ClearConn, DialDgt		
2. D3 completes dialling D2.	**Digits Dialled**			D2's number is 1234
	• diallingConnection	D3C1		
	• diallingDevice	D3		
	• diallingSequence	"1234"		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	none		
	Originated			
	• originatedConnection	D3C1		
	• callingDevice	D3		
	• calledDevice	D1		
	• lastRedirectionDev	NS		
	• localConnectionInfo	connected		
	• cause	normal		
	• servicesPermitted	ClearConn		

Table 5-97 Keyset Call (page 1 of 2)

Activity	Monitored Device D3		Monitored Device D1		Comments
3. D1 is rung	**Delivered**		**Delivered**		
	• deliveredConnection	D1C1	• deliveredConnection	D1C1	
	• alertingDevice	D1	• alertingDevice	D1	
	• callingDevice	D3	• callingDevice	D3	
	• calledDevice	D1	• calledDevice	D1	
	• lastRedirectionDev	NS	• lastRedirectionDev	NS	
	• localConnectionInfo	connected	• localConnectionInfo	alert	
	• cause	normal	• cause	normal	
	• servicesPermitted	ClearConn, Callback, SendUI	• servicesPermitted	ClearConn, Answer, Deflect, SendUI	
4. D2 answers call by going offhook	**Established**		**Established**		
	• estbConnection	D1C1	• estbConnection	D1C1	
	• AnswerringDevice	D1	• AnswerringDevice	D1	
	• callingDevice	D3	• callingDevice	D3	
	• calledDevice	D1	• calledDevice	D1	
	• lastRedirectionDev	NS	• lastRedirectionDev	NS	
	• localConnectionInfo	connected	• localConnectionInfo	connected	
	• cause	normal	• cause	normal	
	• servicesPermitted	ClearConn, Hold, Consultatiol, SST, GenDigits, GenTelTon, SendUI	• servicesPermitted	ClearConn, Hold, Consultation, SST, GenDigits, GenTelTon, SendUI	
5. D2 bridges in by pressing the key for line D1	**Conferenced**		**Conferenced**		Please note that Device D2 does not receive ANY event, because it does not use its own primary line, but bridges on to the primary line of D1.
	• primaryOldCall	D3C1	• primaryOldCall	D1C1	
	• secondaryOldCall	NP	• secondaryOldCall	NP	
	• conferencingDevice	D1	• conferencingDevice	D1	
	• addedDevice	D1	• addedDevice	D1	
	• ConnectionList 1.new / old 2. new 3. new / old	 (D3C2) / (D3C1) (D1C2) (D1C2) ((D1C1)	• ConnectionList 1.new / old 2. new 3. new / old	 (D3C2) / (D3C1) (D1C2) (D1C2) ((D1C1)	
	• localConnectionInfo	connected	• localConnectionInfo	connected	
	• cause	keyOperation	• cause	keyOperation	
	• servicesPermitted	SendUI	• servicesPermitted	SendUI	

Table 5-97 Keyset Call (page 2 of 2)

Remark:

A keyset is a multiline device. CSTA Phase III modells it as independent shared bridge appearances. See ECMA 269 A.2.3. However this is not supported by OpenScape 4000 CSTA V1.

5.19.6 Making Calls in an Executive-Secretary team (CheSe feature)

5.19.6.1 General Remarks

In an Executive-Secretary team calls to the Executive are redirected to a Secretary. The configuration can consist of a maximum of 4 Executives and 2 Secretaries.

The Secretary can have a Representative. When the Secretary activates its Representative ,then all Chese calls will be redirected to the Representative.

It is possible to deactivate the CheSe feature temporarily by either the Executive or the Secretary.

The Chese feature has higher priority in the following situations:

- Call Forward Immediate - if the secretary activates the Call Forward Immediate feature then the forwarding won't take effect, but a normal CheSe call flow happens instead.
- Call during the ParkTimer time - if a third party is parked to the Executive then it is not possible to call the parked-to party, but a basic CheSe call happens instead.

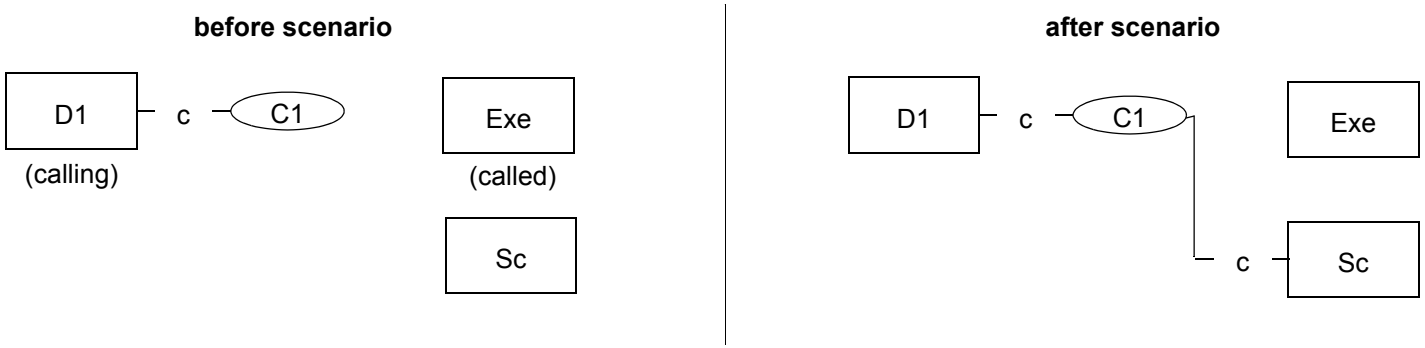
The Chese feature has lower priority in the following situations:

- Secretary and Executive are in the same HG - the normal HG call flow happens.
- Conference - the Executive can be reached from a conference immediately.
- Call Forward Immediate - if the Executive activates the Call Forward Immediate feature then a normal call forwarding happens.

There is a CSTA extension for this feature to make it possible to distinguish between the Secretary's private calls and CheSe calls. A privateData parameter (ExecutiveDeviceID) will be provided in the following events for the Secretary: Delivered, Established, Failed, Queued. This information element contains the extension number of the Executive involved in the CheSe call.

5.19.6.2 Successful basic call - call to Executive

This scenario illustrates the event flow of a successful basic CheSe call. A call comes to the Executive and it is redirected to the Secretary.



See “Manually dialled call”, on page 5-4 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1		Monitored Device Exe	Monitored Device Sc		Comments
1. Secretary starts ringing.	**Delivered**		**none**	**Delivered**		The privateData indicates that it is a Chese call.
	• connection	Sc C1		• connection	Sc C1	
	• alertingDevice	Sc		• alertingDevice	Sc	
	• callingDevice	D1		• callingDevice	D1	
	• calledDevice	Exe		• calledDevice	Exe	
	• lastRedirectionDevice	NK		• lastRedirectionDevice	NK	
	• localConnectionInfo	connected		• localConnectionInfo	alert	
	• cause	forwardImmediate		• cause	forwardImmediate	
	• servicesPermitted	CallBack, ClearConn, SendUserInfo		• servicesPermitted	Answer,ClearConn, Deflect, SendUserInfo	
				• ExecutiveDeviceID	Exe	

Table 5-98 Successful basic call - call to Executive (page 1 of 2)

Activity	Monitored Device D1		Monitored Device Exe	Monitored Device Sc		Comments
2. Secretary answers the call.	**Established** • establishedConnection • answeringDevice • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted	 Sc C1 Sc D1 Exe NS connected normal ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	**none**	**Established** • establishedConnection • answeringDevice • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted • ExecutiveDeviceID	 Sc C1 Sc D1 Exe NS connected normal ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo **Exe**	The privateData indicates that it is a Chese call.

Table 5-98 Successful basic call - call to Executive (page 2 of 2)

Remark:

None

5.19.6.3 Successful basic call - Representative is activated

In case of a “Successful basic call - Representative is activated” similar event flow will be generated to the “Successful basic call - call to Executive” case.

See “Successful basic call - call to Executive”, on page 5-209.

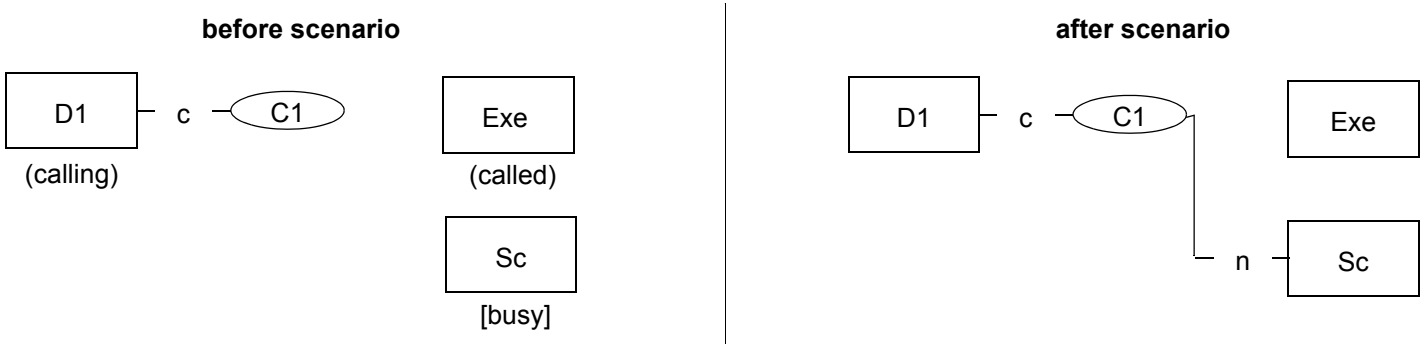
The only difference is that device Sc will be device Rep in this case.

Remark:

The Secretary has activated a preconfigured Representative. See “General Remarks”, on page 5-208.

5.19.6.4 Unsuccessful basic call - Secretary is busy

This scenario illustrates the event flow of an unsuccessful basic CheSe call. A call comes to the Executive and it is re-directed to the Secretary who has another call also for the same Executive.



See “Manually dialled call” on page 5-4 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device Exe	Monitored Device Sc	Comments
1. Secretary is busy. The call can not be completed. D1 hears busy tone.	**Failed** • failedConnection Sc C1 • failingDevice Sc • callingDevice D1 • calledDevice Exe • lastRedirectionDevice NS • localConnectionInfo connected • cause busy • servicesPermitted ClearConn	**none**	**Failed** • failedConnection Sc C1 • failingDevice Sc • callingDevice D1 • calledDevice Exe • lastRedirectionDevice NS • localConnectionInfo fail • cause busy • servicesPermitted ClearConn • ExecutiveDeviceID **Exe**	The privateData indicates that it is a Chese call.
2. The busy connection is cleared immediately.	**Connection Cleared** • droppedConnection Sc C1 • releasingDevice Sc • localConnectionInfo connected • cause normalClr • servicesPermitted ClearConn	**none**	**Connection Cleared** • droppedConnection Sc C1 • releasingDevice Sc • localConnectionInfo null • cause normalClr • servicesPermitted none	

Remark:

None

5.19.6.5 Unsuccessful basic call - Executive is busy

In case of a “Unsuccessful basic call - Executive is busy” similar event flow will be generated to the “Unsuccessful basic call - Secretary is busy” case.

See “Unsuccessful basic call - Secretary is busy”, on page 5-210.

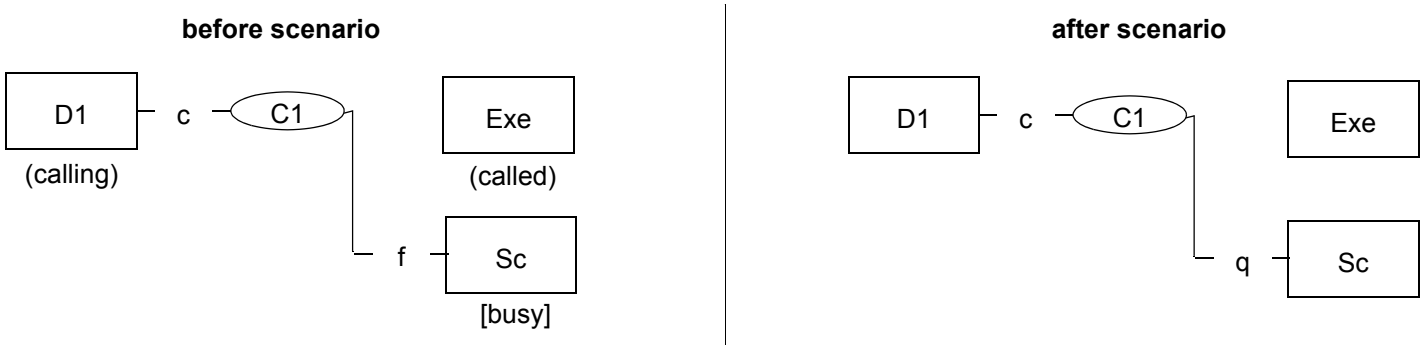
The only difference is that device Exe is busy in this case.

Remark:

Although in this case the Executive is busy, the event flow will be the same.

5.19.6.6 Camp on Executive - Secretary is busy

This scenario illustrates the event flow of an unsuccessful basic CheSe call followed by a manual camp on. A call comes to the Executive and it is redirected to the Secretary who has another call also for the same Executive. After receiving the busy tone, the calling device hits the camp on key.



See “Unsuccessful basic call - Secretary is busy”, on page 5-210 for the event flow to get into the “before scenario” state.

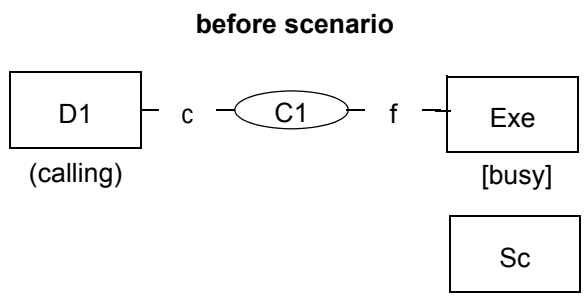
Activity	Monitored Device D1		Monitored Device Exe	Monitored Device Sc		Comments
1. D1 presses the camp on key.	**Queued**		**none**	**Queued**		The privateData indicates that it is a Chese call.
	• queuedConnection	Sc C1		• queuedConnection	Sc C1	
	• queue	Sc		• queue	Sc	
	• callingDevice	D1		• callingDevice	D1	
	• calledDevice	Exe		• calledDevice	Exe	
	• lastRedirectionDevice	NS		• lastRedirectionDevice	NS	
	• localConnectionInfo	connected		• localConnectionInfo	queued	
	• cause	campOn		• cause	campOn	
	• servicesPermitted	CallBack, ClearConn, SendUserInfo		• servicesPermitted	SendUserInfo	
				• ExecutiveDeviceID	**Exe**	

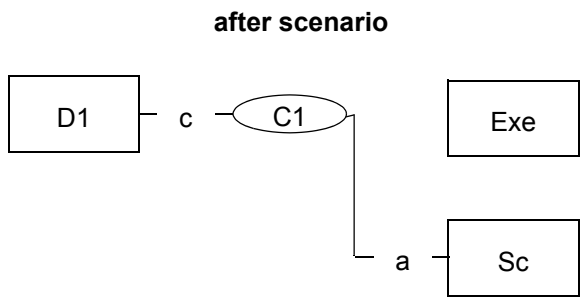
Remark:

After the Queued event no privateData is present in the events of device Sc due to the limitation of the switch.

5.19.6.7 Camp on Executive - Executive is busy

This scenario illustrates the event flow of an unsuccessful basic CheSe call followed by a manual camp on. A call comes to the Executive who has another call. After receiving the busy tone, the calling device hits the camp on key. The call is redirected to the Secretary.





See “Unsuccessful basic call - Executive is busy”, on page 5-212 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1		Monitored Device Exe	Monitored Device Sc		Comments
1. Secretary starts ringing.	**Delivered** • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted	 Sc C1 Sc D1 Exe NK connected forwardImmediate CallBack, ClearConn, SendUserInfo	**none**	**Delivered** • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • localConnectionInfo • cause • servicesPermitted • ExecutiveDeviceID	 Sc C1 Sc D1 Exe NK alert forwardImmediate Answer,ClearConn, Deflect, SendUserInfo Exe	The privateData indicates that it is a Chese call.

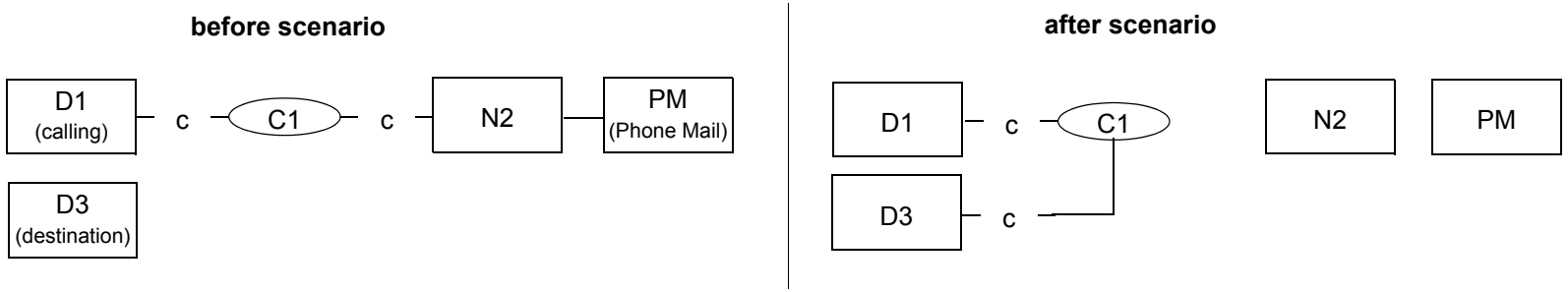
Remark:

None

5.19.7 Single Step Call Transfer with Await Connect

5.19.7.1 Basic successful SSCT call with Await Connect

This scenario describes a call flow when Phone Mail transfers with the Await Connect feature.



See “Internal ACD call completed to Phone Mail agent” on page 5-117 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device N2	Monitored Device D3	Comments
1. D3 is promted to go offhook.			**Service Initiated** • initiatedConnection D3C2 • initiatingDevice D3 • localConnectionInfo initiated • cause transfer • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo • privateEventCause SSCT	The SSCT private event cause shows, that this feature has been used.
2. Device D3 goes offhook.			**Connection Cleared** • droppedConnection D3C2 • releasingDevice D3 • localConnectionInfo null • cause normalClr • servicesPermitted none • privateEventCause SSCT	The SSCT private event cause shows, that the Connection Cleared does not really mean, that D3 is idle. The next event will be an Established on this monitor.
3. Phone Mail transfers.	**Transferred** • primaryOldCall D1C1 • transferringDevice PM • transferredToDevice D3 • transferredConnections 1.new (D1C1) 2.new (D3C1) • localConnectionInfo connected • cause SST • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	**Transferred** • primaryOldCall N2C1 • transferringDevice PM • transferredToDevice D3 • transferredConnections 1.new (D1C1) 2.new (D3C1) • localConnectionInfo null • cause SST • servicesPermitted none		OpenScape 4000 modells a SSCT as a Single Step Transfer, that is why this cause is sent with the Tramsferred event.

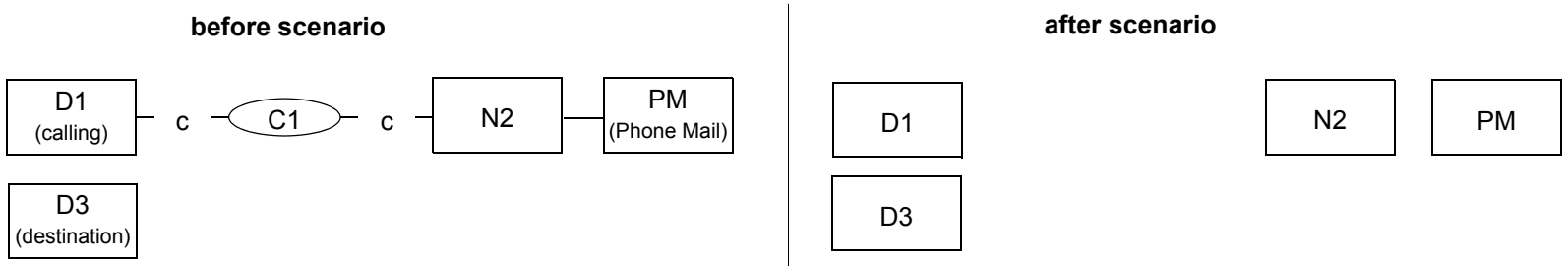
Table 5-99 Single Step Call Transfer (Await connect) (page 1 of 2)

Activity	Monitored Device D1	Monitored Device N2	Monitored Device D3	Comments
4. The connection is immediately estabilished.	**Established** • establishedConnection D3C1 • answeringDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice PM • localConnectionInfo connected • cause SST • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTone, SendUserInfo		**Established** • establishedConnection D3C1 • answeringDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice PM • localConnectionInfo connected • cause SST • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTone, SendUserInfo	OpenScape 4000 modells a SSCT as a Single Step Transfer, that is why this cause is sent with the Established event.

Table 5-99 Single Step Call Transfer (Await connect) (page 2 of 2)

5.19.7.2 Unsuccessful basic SSCT call with Await Connect

This scenario describes a call flow when Phone Mail tries to transfers with the Await connect feature, but the calling party hangs up.



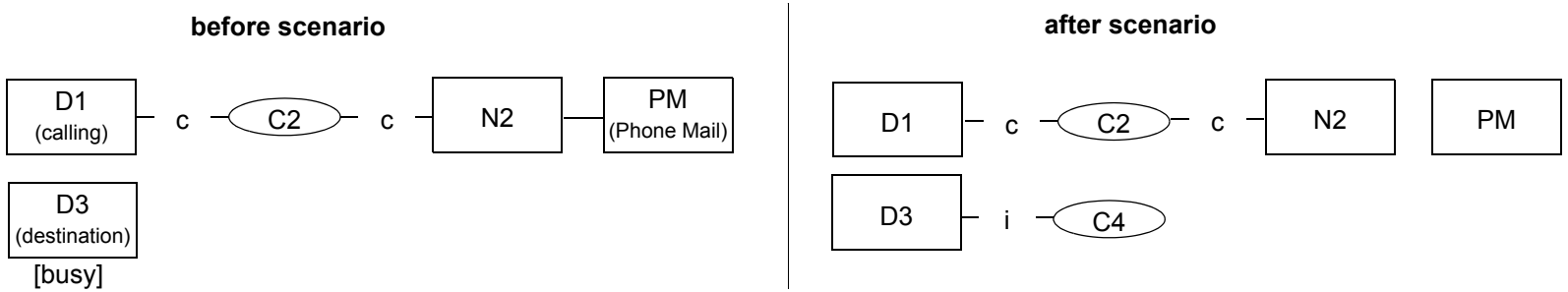
See “Internal ACD call completed to Phone Mail agent” on page 5-117 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device N2	Monitored Device D3	Comments
1. D3 is promted to go offhook.			**Service Initiated** • initiatedConnection D3C2 • initiatingDevice D3 • localConnectionInfo initiated • cause transfer • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo • privateEventCause SST	
2. Device D1 goes onhook. This results in a D3C2 clearing.			**Connection Cleared** • droppedConnection D3C2 • releasingDevice D3 • localConnectionInfo null • cause normalClr • servicesPermitted none • privateEventCause notPresent	
3. Device D1 goes onhook.	**Connection Cleared** • droppedConnection D1C1 • releasingDevice D1 • localConnectionInfo null • cause normalClr • servicesPermitted none	**Connection Cleared** • droppedConnection D1C1 • releasingDevice D1 • localConnectionInfo connected • cause normalClr • servicesPermitted ClearConn		
4. The remaining connection is cleared.		**Connection Cleared** • droppedConnection N2C1 • releasingDevice N2 • localConnectionInfo null • cause normalClr • servicesPermitted none		

Table 5-100 Unsuccessful Single Step Call Transfer (Await connect)

5.19.7.3 SSCT call with Await Connect , Camp On

This scenario describes a call flow when Phone Mail transfers with the Await connect feature. Destination is non-idle but Camp On is possible.



See “Internal ACD call completed to Phone Mail agent” on page 5-117 for the event flow to get into the “before scenario” state.

Activity	Monitored Device D1	Monitored Device N2	Monitored Device D3	Comments
1. The SSCT call camps onto Device D3.			**Queued** • queuedConnection D3C3 • queue D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NS • localConnectionInfo queued • cause campon • servicesPermitted SendUserInfo • privateEventCause SST	
2. Device D3 clears itself from its previous call.			**Connection Cleared** • droppedConnection D3C1 • releasingDevice D3 • localConnectionInfo null • cause normalClr • servicesPermitted none	C1 was the active call of D3.
3. D3 clears the camped connection.			**Connection Cleared** • droppedConnection D3C3 • releasingDevice D3 • localConnectionInfo null • cause normalClr • servicesPermitted none	

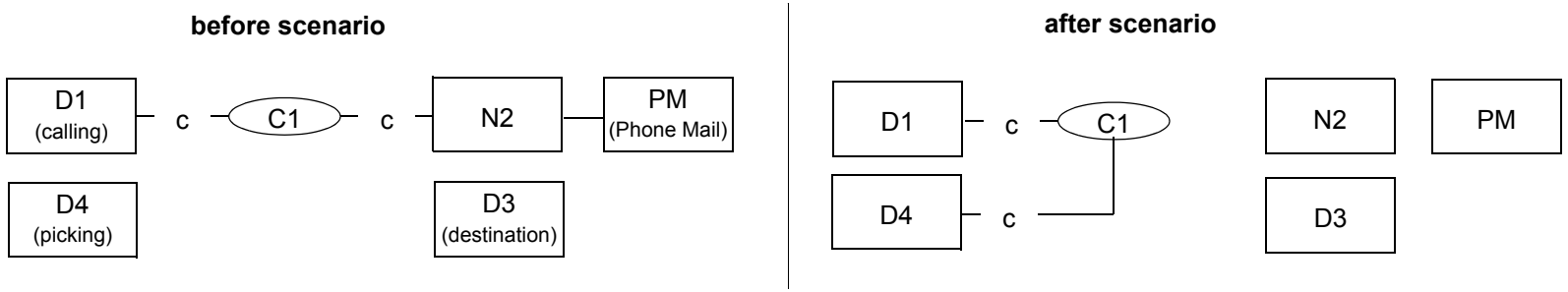
Table 5-101 Single Step Call Transfer, Camp On (page 1 of 2)

Activity	Monitored Device D1	Monitored Device N2	Monitored Device D3		Comments
4. D3 is promted to go offhook.			**Service Initiated**		
			• initiatedConnection	D3C2	
			• initiatingDevice	D3	
			• localConnectionInfo	initiated	
			• cause	transfer	
			• servicesPermitted	Answer, ClearConn, Deflect, SendUserInfo	
			• privateEventCause	SST	

Table 5-101 Single Step Call Transfer, Camp On (page 2 of 2)

5.19.7.4 SSCT call with Await Connect , Pick Up

This scenario describes a call flow when Phone Mail tries to transfers with the Await connect feature. A device in the destinations's group picks the call up.



See "Internal ACD call completed to Phone Mail agent" on page 5-117 for the event flow to get into the "before scenario" state.

Activity	Monitored Device D1	Monitored Device N2	Monitored Device D3	Monitored Device D4	Comments
1. D3 is promted to go offhook.			**Service Initiated** • initiatedConnection D3C2 • initiatingDevice D3 • localConnectionInfo initiated • cause transfer • servicesPermitted Answer, ClearConn, Deflect, SendUserInfo • privateEventCause SST		
2. D4 goes offhook.				**Service Initiated** • initiatedConnection D4C3 • initiatingDevice D4 • localConnectionInfo initiated • cause normal • servicesPermitted ClearConn, DialDgt	
3. Device D4 picks.			**Connection Cleared** • droppedConnection D3C2 • releasingDevice D3 • localConnectionInfo null • cause normalClr • servicesPermitted none • privateEventCause notPresent		
4. Device D4 clears the call from which it picked up.				**Connection Cleared** • droppedConnection D4C3 • releasingDevice D4 • localConnectionInfo null • cause normalClr • servicesPermitted none	

Table 5-102 Single Step Call Transfer, Pick Up (page 1 of 2)

Activity	Monitored Device D1	Monitored Device N2	Monitored Device D3	Monitored Device D4	Comments
5. Phone Mail transfers.	**Transferred** • primaryOldCall D1C1 • transferringDevice PM • transferredToDevice D4 • transferredConnections (D1C1) (D4C1) 1.new 2.new • localConnectionInfo conn • cause SST • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	**Transferred** • primaryOldCall N2C1 • transferringDevice PM • transferredToDevice D4 • transferredConnections (D1C1) (D4C1) 1.new 2.new • localConnectionInfo null • cause SST • servicesPermitted none			
6. The connection is immediately established.	**Established** • establishedConnection D4C1 • answeringDevice D4 • callingDevice D1 • calledDevice D4 • lastRedirectionDevice PM • localConnectionInfo connected • cause pickup • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTone, SendUserInfo • assocCalledDevice N1			**Established** • establishedConnection D4C1 • answeringDevice D4 • callingDevice D1 • calledDevice D4 • lastRedirectionDevice PM • localConnectionInfo connected • cause pickup • servicesPermitted ClearConn, Consult, Hold, SST, GenDgt, GenTelTone, SendUserInfo • assocCalledDevice N1	

Table 5-102 Single Step Call Transfer, Pick Up (page 2 of 2)

5.19.8 Concept of "presentation indicator for devices" in CSTA events

To provide adaptable solution for every application, CA4000 will have three different solution to handle the presentation indicator for devices. The different solutions can be activated in the Advanced Configuration page of the Connectivity Adapter. The parameter PRESENTATION_RESTRICTED should be set to the following values:

- "normal" : to provide the old concept as it worked in the past (valid for CSTA III)
- "ignore" : to have the presentation restricted **partially ignored** (valid for CSTA III); this is **only a work around** for OpenScape ProCenter, which should also use "private data" from now on
- "private data" : presentation indicator will be represented by private data (valid only for CSTA III interface)
  e.g.:PRESENTATION_RESTRICTED = private data
- "extended private data": a new concept of the presentation indicator introduced in HiPath4000 V4.0 and V5.0 whith that the switch delivers presentation restricted information not only in case of calling and called party. All presentation indicator will be represented by private data (valid only for CSTA III interface)
  PRESENTATION_RESTRICTED=private data
  ALLOW_ALL_PRIVATE_DATA=True
- "special": similar to the functionality of "normal" but provides possibility for the OpenScape ContactCenter (special customer change request for Bundestag) to replace the "not Known" with the given <special value>
  PRESENTATION_RESTRICTED=special
  PRESENTATION_RESTRICTED_SPECIAL_VALUE=<special value>

The application has the choice when to switch-over to one of the "private data". By default the parameter PRESENTATION_RESTRICTED and ALLOW_ALL_PRIVATE_DATA are not included in the configurable parameters in the Connectivity Adapter's Advanced Configuration page.

If these parameters are not in the config file then the "normal" behaviour will be activated automatically. CA4000 reads the parameter in case of start or restart, so any change will be valid only after the **start or restart**.

Remark: Presentation restricted info is not stored in CA4000 due to consistency considerations. Therefore OpenScape 4000 CSTA can only handle devices as secret, if the switch informs it explicitly. That means the device might not signaled as restricted, if it was signaled only earlier.
In example if secret party A calls B, B does not answer the call, and a ForwardNoAsnwer triggers and the call is diverted to C, then A will be signaled in the DIVERTED event as visible.

5.19.8.1 The old concept of presentation indicator (“normal”)

This concept handles the private data as it was handled in the past. It means that if ACL indicates either the calling or the called party presentation restricion field then CA4000 will report these devices as “Not Known” in the CSTA events. This behaviour is valid for both the CSTA III interface. The following scenarios describe this behaviour:

5.19.8.2 Basic Internal Call with presentation restricted devices (CSTA III)

The scenario describes a basic internal call scenario when both devices have presentation restricton.

The parties in this call scenario are:

- Party A: station (such as a digital or analog telephone) (presentation restricted)
- Party B: station (presentation restricted)

Activity	Monitored Device D1		Monitored Device D2
1. D1 goes off-hook.	**Service Initiated**		
	• initiatedConnection	NP, C1	
	• initiatingDevice	NK	
	• localConnectionInfo	initiated	
	• cause	normal	
	• servicesPermitted	ClearConn, DialDgt	
2. D1 completes dialling D2.	**Digits Dialled**		
	• diallingConnection	NP, C1	
	• diallingDevice	NK	
	• diallingSequence	“1234“	
	• localConnectionInfo	initiated	
	• cause	normal	
	• servicesPermitted	none	
	Originated		
	• originatedConnection	NP, C1	
	• callingDevice	NK	
	• calledDevice	D2	
	• localConnectionInfo	connected	
	• cause	normal	
	• servicesPermitted	ClearConn	

Activity	Monitored Device D1		Monitored Device D2	
3. D2 starts ringing.	**Delivered**		**Delivered**	
	• connection	NP,C1	• connection	NP,C1
	• alertingDevice	NK	• alertingDevice	NK
	• callingDevice	NK	• callingDevice	NK
	• calledDevice	NK	• calledDevice	NK
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS
	• localConnectionInfo	connected	• localConnectionInfo	alert
	• cause	normal	• cause	normal
	• servicesPermitted	CallBack, ClearConn, SendUserInfo	• servicesPermitted	AnswerCall, ClearConn, Deflect, SendUserInfo
4. D2 answers the call.	**Established**		**Established**	
	• establishedConnection	NP,C1	• establishedConnection	NP,C1
	• answeringDevice	NK	• answeringDevice	NK
	• callingDevice	NK	• callingDevice	NK
	• calledDevice	NK	• calledDevice	NK
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS
	• localConnectionInfo	connected	• localConnectionInfo	connected
	• cause	normal	• cause	normal
	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo

5.19.8.3 Blind Transfer

The following table shows a call scenario where blind transfer of call is made.

The parties in these call scenarios are:

- Party A: held (presentation restricted)
- Party B: consulting (presentation restricted)
- Party C: consulted (presentation restricted)

Telephony Activity	Event (Party A)	Event (Party B)	Event (Party C)
Party A and Party B are in conversation. 1. Party B presses the consultation/ conference key.	**Held** • CID: 1, NP • Holding DID: NK • LCS: connected • Private cause: consultation hold	**Held** • CID: 1, B • Holding DID: B • LCS: held • Private cause: consultation hold **Service Initiated** • CID: 2, B • LCS: initiated	None

Table 5-103 Blind Transfer – Consulted Party Answers (page 1 of 3)

Telephony Activity	Event (Party A)	Event (Party B)	Event (Party C)
2. Party B dials and rings Party C.	None	**Delivered** • CID: 2, NP • Alerting DID: NK • Calling DID: NK • Originally called DID: NK • Last redirecting DID: NR • LCS: connected	**Delivered** • CID: 2, NP • Alerting DID: NK • Calling DID: NK • Originally called DID: NK • Last redirecting DID: NR • LCS: alerting • Private consultation held with party: A
3. Party B transfers by going on-hook.	**Transferred** • Primary old CID: 1, B • Transferring DID: B • Transferred DID: NK • Transferred CID: (3, A); (3,NP) • LCS: connected	**Transferred** • Primary old CID: 1, NP • Secondary old CID: 2, B • Transferring DID: NK • Transferred DID: C • Transferred CIDs: (3, C); (3, A) • LCS: null	**Transferred** • Primary old CID: 2, B • Transferring DID: B • Transferred DID: C • Transferred CIDs: (3, C);(3, NP) • LCS: alerting

Table 5-103 Blind Transfer – Consulted Party Answers (page 2 of 3)

Telephony Activity	Event (Party A)	Event (Party B)	Event (Party C)
4. Party C answers.	**Established** • CID: 3, NP • Answering DID: NK • Calling DID: NK • Originally called DID: NK • Last redirecting DID: NR • LCS: connected	None	**Established** • CID: 3, NP • Answering DID: NK • Calling DID: NK • Originally called DID: NK • Last redirecting DID: NR • LCS: connected

Table 5-103 Blind Transfer – Consulted Party Answers (page 3 of 3)

5.19.8.4 Conference Initiation by Conference Master with presentation restricted devices

Party A calls Party B and is connected (call ID 1). Party B presses the consultation key and calls Party C (call ID 2). Party A is held. Party B and C are connected.

The parties in this call scenario are

- Party A: held party (presentation restricted)
- Party B: consulting (presentation restricted)
- Party C: consulted (presentation restricted)

Telephony Activity	Event (Party A)	Event (Party B)	Event (Party C)
1. Party B initiates a 3-party conference, pressing the consultation/ conference key or requesting the Conference Call service.	**Conferenced** • Primary old CID: 1, NP • Conference controller: NK • Added DID: C • Conference CIDs: (3,C) • LCS: connected	**Conferenced** • Primary old CID: 1, B • Secondary old CID: 2, B • Conference controller: B • Added DID: NK • Conference CIDs: (3,A) • LCS: connected	**Conferenced** • Primary old CID: 2, NP • Conference controller: NK • Added DID: C • Conference CIDs: (3,A) • LCS: connected

Table 5-104 Conference Initiation by Conference Master

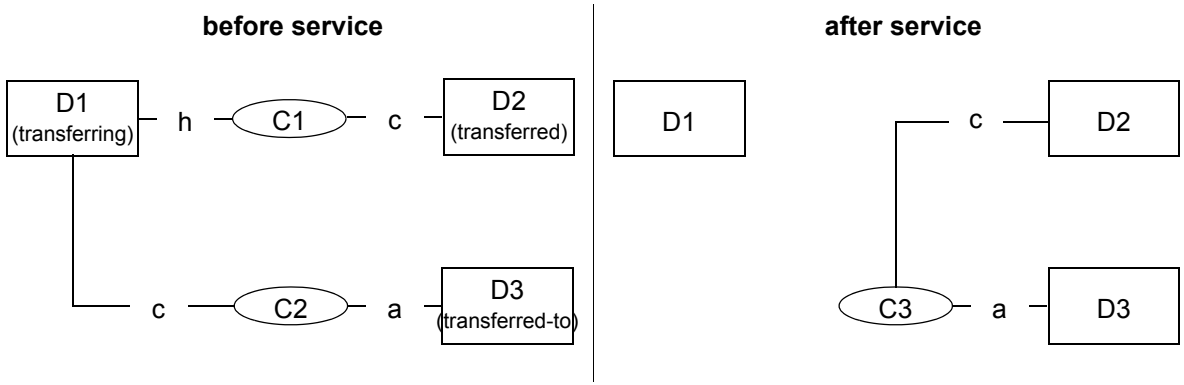
5.19.8.5 Blind Transfer with presentation restricted devices (CSTA III)

This service transfers a held party to a consulted party. The transfer service request is issued before the consulted device connects into the new call.

Device D1: presentation restricted

Device D2: presentation restricted

Device D3: presentation restricted



See “Succesful consultation call” for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Transfer Call service is invoked on behalf of device D1.	**Transfer Call Request** • heldConnection D1C1 • activeConnection D1C2			
2. Acknowledgement.	**Transfer Call Response** • transferredConnection D3C3			

Table 5-105 Blind Transfer with presentation restricted devices (page 1 of 2)

Activity	Monitored Device D1		Monitored Device D2		Monitored Device D3		Comments
3. Calls between D1, D2 and D1, D3 are released. The connections between D2, D1 and D3, D1 are replaced with a single connection between D2 and D3.	**Transferred**		**Transferred**		**Transferred**		The CSTA Transferred event Local View modeling option is provided by the switching function. This means that the primary old call parameters in the Transferred event represent a device oriented view.
	• primaryOldCall	D1C1	• primaryOldCall	D2C1	• primaryOldCall	D3C2	
	• secondaryOldCall	D1C2					
	• transferringDevice	NotKnown	• transferringDevice	D1	• transferringDevice	D1	
	• transferredToDevice	D3	• transferredToDevice	NotKnown	• transferredToDevice	NotKnown	
	• transferredConnections		• transferredConnections		• transferredConnections		
	1. new / old	(D2C3) / (D2C1)	1. new / old	(C3) / (C1)	1. new / old	(D3C3) / (D3C2)	
	2. new / old	(D3C3) / (D3C2)	2. new	(D3C3)	2. new	(C3)	
	• localConnectionInfo	null	• localConnectionInfo	connected	• localConnectionInfo	alerting	
	• cause	Transfer	• cause	Transfer	• cause	Transfer	
	• servicesPermitted	none	• servicesPermitted	ClearConn, SendUserInfo	• servicesPermitted	Answer, ClearConn, SendUserInfo	

Table 5-105 Blind Transfer with presentation restricted devices (page 2 of 2)

Remark:

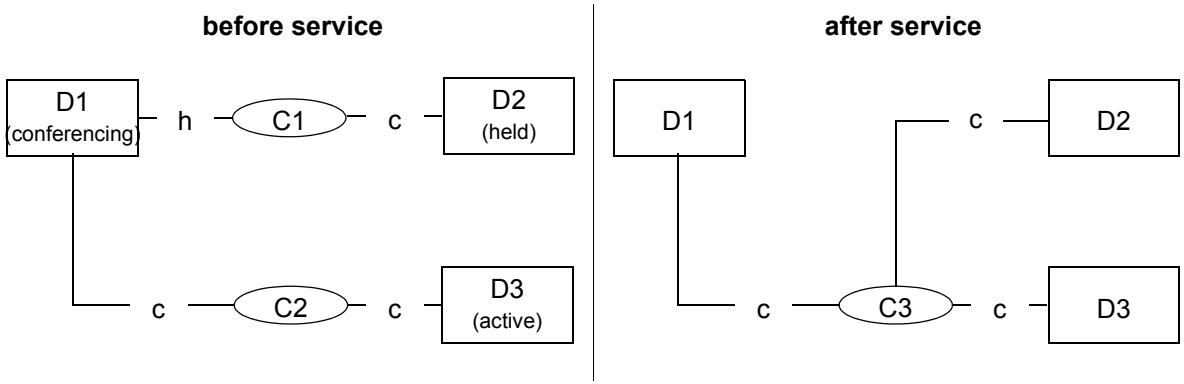
5.19.8.6 Conference with presentation restricted devices (CSTA III)

This service provides a conference of an existing held call and another active call at a conferencing device. The two calls are merged into a single call at the conferencing device.

Device D1: presentation restricted

Device D2: presentation restricted

Device D3: presentation restricted



See “Successful consultation call” for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Conference Call service is requested on behalf of device D1.	**Conference Request** • heldConnection D1C1 • activeConnection D1C2			
2. Acknowledgement.	**Conference Response** • conferencedConnection D1C3			

Table 5-106 Conference with presentation restricted devices (page 1 of 2)

Activity	Monitored Device D1		Monitored Device D2		Monitored Device D3		Comments
3. Conference estabilished.	**Conferenced**		**Conferenced**		**Conferenced**		The addedParty specifies the device ID of the device, that belongs to the active (not held) call of the conference. Note that the primaryOldCall and the secondaryOldCall parameters follows the "local view" modeling option.
	• primaryOldCall	D1C1	• primaryOldCall	D2C1	• primaryOldCall	D3C2	
	• secondaryOldCall	D1C2					
	• conferencingDevice	NotKnown	• conferencingDevice	NotKnown	• conferencingDevice	NotKnown	
	• Added	NotKnown	• Added	D3	• Added	NotKnown	
	• conferenceConnections		• conferenceConnections		• conferenceConnections		
	1. new/old	(D1C3)/(D1C1)	1. new/old	(D2C3)/(D2C1)	1. new/old	(D1C3)/(D1C2)	
	2. new/old	(D1C3)/(D1C2)	2. new/old		2. new/old		
	3. new	(D2C3)	3. new	(D3C3)	3. new	(D2C3)	
	4. new	(D3C3)					
	• localConnectionInfo	connected	• localConnectionInfo	connected	• localConnectionInfo	connected	
	• cause	normal	• cause	normal	• cause	normal	
	• servicesPermitted	ClearConn, Consult, Hold, SendUserInfo	• servicesPermitted	ClearConn, Consult, Hold, SendUserInfo	• servicesPermitted	ClearConn, Consult, Hold, SendUserInfo	

Table 5-106 Conference with presentation restricted devices (page 2 of 2)

Remark:

The manual case is similar to the described event flow.

5.19.8.7 Presentation restricted is ignored

This solution was initially developed for the application Procenter. The restriction of the presentation will be partially ignored. It means that the devices with presesntation restriction will be reported in the CSTA III events.

In case of the call leaves the swithcing subdomain (incoming and outgoing calls) the restricted parameters will be reported as "Not Known" althought the trunk provides the network party. It means that if a trunk information contains the network party field but its presentation indicator is restricted then CA4000 will not provide this number! The application developer has to take it into consideration when he/she selects this solution.

5.19.8.8 Presentation indicator represented by Private Data

The presentation indicator of a device indicates whether the dialling number of a device is allowed or restricted. In case of any restricted presentation reported in the calling or called field of an ACL event CA4000 will provide a private data called Presentation Restricted Device 1 or Presentation Restricted Device 2.

The presentationRestrictedDeviceID1/2 refers to the CSTA calling/called device, that means CA4000 provides restriction information only for the calling, called party. However CA4000 provides restriction information in specific cases for those events (Transferred, Conference, etc), where the CSTA calling/called device do not exist. Due to ACL limitations, CA4000 cannot provide proper restriction information in these cases, but applications can use this additional information as it is.

Remark: Te new special extended OpenScape4000 concept of the presentation indicator can provide infromation about secret devices without any limitation. Please activate the PRESENTATION_RESTRICTED=private data ALLOW_ALL_PRIVATE_DATA=True in the configuration file and restart the CA4000.

The following scenarios describe the private data representation behavour. (Old concept!)

5.19.8.9 Illustration of the new concept:

In the following scenarios the presentation indicator of Party A and Party B is restricted.

1. Party A calls party B.

Delivered Event:

Calling: A

Called: B

presentationRestrictedDeviceID1: A

presentationRestrictedDeviceID2: B

2. Party A on node1 calls party B on node2.

Delivered Event on monitored party B:

Calling: A

Called: B

presentationRestrictedDeviceID1: A

presentationRestrictedDeviceID2: B

3.Party A on node1 calls party B on node2.

Delivered Event on monitored party A:

Calling: A

Called: B

presentationRestrictedDeviceID1: A

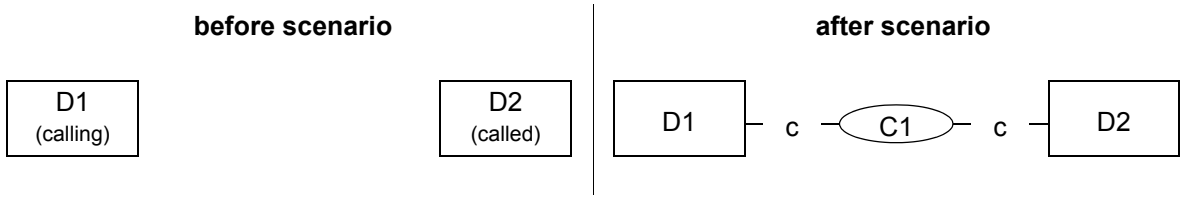
presentationRestrictedDeviceID2: -

5.19.8.10 Basic call with presentation restricted devices

This scenario illustrates a call originated through manual device activity.

Device D1: presentation restricted

Device D2: presentation restricted



Activity	Monitored Device D1		Monitored Device D2	Comments
1. D1 goes off-hook.	**Service Initiated**			
	• initiatedConnection	D1C1		
	• initiatingDevice	D1		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	ClearConn, DialDgt		
	• Presentation Restricted Device1	D1		
2. D1 completes dialling D2.	**Digits Dialled**			D2's number is 1234
	• diallingConnection	D1C1		
	• diallingDevice	D1		
	• diallingSequence	"1234"		
	• localConnectionInfo	initiated		
	• cause	normal		
	• servicesPermitted	none		
	Originated			
	• originatedConnection	D1C1		
	• callingDevice	D1		
	• calledDevice	D2		
	• localConnectionInfo	connected		
	• cause	normal		
	• servicesPermitted	ClearConn		
	• Presentation Restricted Device1	D1		

Table 5-107 Basic call with presentation restricted devices (page 1 of 3)

Activity	Monitored Device D1		Monitored Device D2		Comments
3. D2 starts ringing.	**Delivered**		**Delivered**		
	• connection	D2C1	• connection	D2C1	
	• alertingDevice	D2	• alertingDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	alert	
	• cause	normal	• cause	normal	
	• servicesPermitted	CallBack, ClearConn, SendUserInfo	• servicesPermitted	AnswerCall, ClearConn, Deflect, SendUserInfo	
	• Presentation Restricted Device1	D1	• Presentation Restricted Device1	D1	
	• Presentation Restricted Device2	D2	• Presentation Restricted Device2	D2	
4. D2 answers the call.	**Established**		**Established**		
	• establishedConnection	D2C1	• establishedConnection	D2C1	
	• answeringDevice	D2	• answeringDevice	D2	
	• callingDevice	D1	• callingDevice	D1	
	• calledDevice	D2	• calledDevice	D2	
	• lastRedirectionDevice	NS	• lastRedirectionDevice	NS	
	• localConnectionInfo	connected	• localConnectionInfo	connected	
	• cause	normal	• cause	normal	
	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	• servicesPermitted	ClearConn, Consult, Hold, SST, GenDgt, GenTelTones, SendUserInfo	
	• Presentation Restricted Device1	D1	• Presentation Restricted Device1	D1	
	• Presentation Restricted Device2	D2	• Presentation Restricted Device2	D2	
5. D2 goes on-hook.	**Connection Cleared**		**Connection Cleared**		
	• droppedConnection	D2C1	• droppedConnection	D2C1	
	• releasingDevice	D2	• releasingDevice	D2	
	• localConnectionInfo	connected	• localConnectionInfo	null	
	• cause	normalClr	• cause	normalClr	
	• servicesPermitted	ClearConn	• servicesPermitted	none	
	• Presentation Restricted Device1	D1	• Presentation Restricted Device1	D2	

Table 5-107 Basic call with presentation restricted devices (page 2 of 3)

Activity	Monitored Device D1		Monitored Device D2	Comments
6. The remaining connection D1C1 goes blocked.	**Failed**			
	• failedConnection	D1C1		
	• failingDevice	D1		
	• callingDevice	D1		
	• calledDevice	D2		
	• lastRedirectionDevice	NS		
	• localConnectionInfo	fail		
	• cause	blocked		
	• servicesPermitted	ClearConn		
	• Presentation Restricted Device1	D1		
7. D1 goes on-hook.	**Connection Cleared**			
	• droppedConnection	D1C1		
	• releasingDevice	D1		
	• localConnectionInfo	null		
	• cause	normalClr		
	• servicesPermitted	none		
	• Presentation Restricted Device1	D1		

Table 5-107 Basic call with presentation restricted devices (page 3 of 3)

Remark:

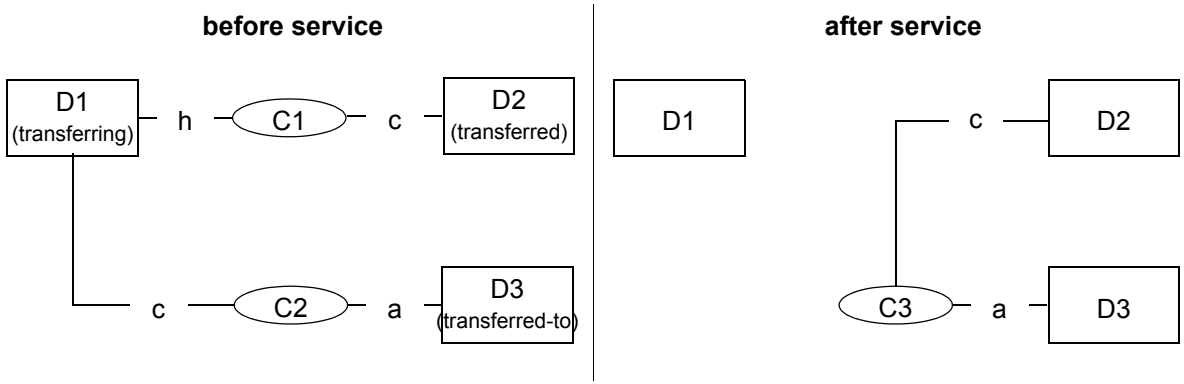
5.19.8.11 Blind Transfer with presentation restricted devices

This service transfers a held party to a consulted party. The transfer service request is issued before the consulted device connects into the new call.

Device D1: presentation restricted

Device D2: presentation restricted

Device D3: presentation restricted



See “Successful consultation call” for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Transfer Call service is invoked on behalf of device D1.	**Transfer Call Request** • heldConnection D1C1 • activeConnection D1C2			
2. Acknowledgement.	**Transfer Call Response** • transferredConnection D3C3			

Table 5-108 Blind Transfer with presentation restricted devices (page 1 of 2)

Activity	Monitored Device D1		Monitored Device D2		Monitored Device D3		Comments
3. Calls between D1, D2 and D1, D3 are released. The connections between D2, D1 and D3, D1 are replaced with a single connection between D2 and D3.	**Transferred**		**Transferred**		**Transferred**		The CSTA Transferred event Local View modeling option is provided by the switching function. This means that the primary old call parameters in the Transferred event represent a device oriented view.
	• primaryOldCall	D1C1	• primaryOldCall	D2C1	• primaryOldCall	D3C2	
	• secondaryOldCall	D1C2					
	• transferringDevice	D1	• transferringDevice	D1	• transferringDevice	D1	
	• transferredToDevice	D3	• transferredToDevice	D3	• transferredToDevice	D3	
	• transferredConnections 1. new / old 2. new / old	 (D2C3) / (D2C1) (D3C3) / (D3C2)	• transferredConnections 1. new / old 2. new	 (D2C3) / (D2C1) (D3C3)	• transferredConnections 1. new / old 2. new	 (D3C3) / (D3C2) (D2C3)	
	• localConnectionInfo	null	• localConnectionInfo	connected	• localConnectionInfo	alerting	
	• cause	Transfer	• cause	Transfer	• cause	Transfer	
	• servicesPermitted	none	• servicesPermitted	ClearConn, SendUserInfo	• servicesPermitted	Answer, ClearConn, SendUserInfo	
	• Presentation Restricted Device1	D1	• Presentation Restricted Device1	D2	• Presentation Restricted Device1	D2	
			• Presentation Restricted Device2	D3	• Presentation Restricted Device2	D3	

Table 5-108 Blind Transfer with presentation restricted devices (page 2 of 2)

Remark:

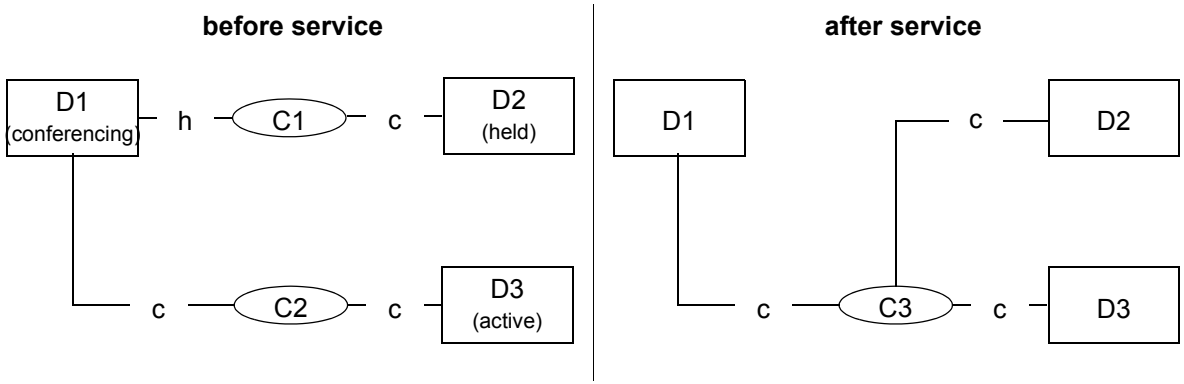
5.19.8.12 Conference with presentation restricted devices

This service provides a conference of an existing held call and another active call at a conferencing device. The two calls are merged into a single call at the conferencing device.

Device D1: presentation restricted

Device D2: presentation restricted

Device D3: presentation restricted



See “Successful consultation call” for the event flow to get into the “before service” state.

Activity	Monitored Device D1	Monitored Device D2	Monitored Device D3	Comments
1. Conference Call service is requested on behalf of device D1.	**Conference Request** • heldConnection D1C1 • activeConnection D1C2			
2. Acknowledgement.	**Conference Response** • conferencedConnection D1C3			

Table 5-109 Conference with presentation restricted devices (page 1 of 2)

Activity	Monitored Device D1		Monitored Device D2		Monitored Device D3		Comments
3. Conference estabilished.	**Conferenced**		**Conferenced**		**Conferenced**		The addedParty specifies the device ID of the device, that belongs to the active (not held) call of the conference. Note that the primaryOldCall and the secondaryOldCall parameters follows the "local view" modeling option.
	• primaryOldCall	D1C1	• primaryOldCall	D2C1	• primaryOldCall	D3C2	
	• secondaryOldCall	D1C2					
	• conferencingDevice	D1	• conferencingDevice	D1	• conferencingDevice	D1	
	• Added	D3	• Added	D3	• Added	D3	
	• conferenceConnections		• conferenceConnections		• conferenceConnections		
	1. new/old	(D1C3)/(D1C1)	1. new/old	(D2C3)/(D2C1)	1. new/old	(D1C3)/(D1C2)	
	2. new/old	(D1C3)/(D1C2)	2. new/old	(D1C3)/(D1C1)	2. new/old	(D3C3)/(D3C2)	
	3. new	(D2C3)	3. new	(D3C3)	3. new	(D2C3)	
	4. new	(D3C3)					
	• localConnectionInfo	connected	• localConnectionInfo	connected	• localConnectionInfo	connected	
	• cause	normal	• cause	normal	• cause	normal	
	• servicesPermitted	ClearConn, Consult, Hold, SendUserInfo	• servicesPermitted	ClearConn, Consult, Hold, SendUserInfo	• servicesPermitted	ClearConn, Consult, Hold, SendUserInfo	
	• Presentation Restricted Device1	D1	• Presentation Restricted Device1	D2	• Presentation Restricted Device1	D1	
	• Presentation Restricted Device2	D3	• Presentation Restricted Device2	D1	• Presentation Restricted Device2	D3	

Table 5-109 Conference with presentation restricted devices (page 2 of 2)

Remark:

The manual case is similar to the described event flow.

5.19.8.13 Affected events

The following events can be **affected** by the new private data elements:

Conferenced, Connection Cleared, Delivered, Diverted, Established, Failed, Held, Network Reached, Originated, Queued, Retrieved, Service Initiated, Transferred, Callback

The new private data elements **will not** be provided in Logical Device Feature events:

Agent Busy, Agent Ready, Agent Not Ready, Agent Working After Call, Agent Logon, Agent Logoff

In the corresponding ACL events there is no ACL calling or called party which could determine the requested private datas.

5.19.8.14 Remarks

5.19.8.15 Multiple calls

Caller ID blocking/unblocking is supported for multiple calls.

It means that the restriction is always in relation to a call. E.g. if a party goes out to consultation (with call ID 2) and activates caller ID blocking only for the consulted call (call ID 2), in the events for the original call (call ID 1) this party will not have name and number restriction, but for the consultation call (call ID 2) this party will have name and/or number restriction.

5.19.8.16 Configuration of presentation indicator

The ways of configuring caller ID blocking for an extension:

5.19.8.17 Configure the presentation indicator by AMO

Set

AMO CHA-SBCSU:..., SSTNO=YES (secret station number)

The party will be restricted in the ACL events for all the incoming and outgoing calls (CallingParty and CalledParty).

5.19.8.18 Configure the presentation indicator via OptiSet menu

Set Service menu -> More features -> Display suppress on.

The party will be restricted only in the forthcoming call (CallingParty)

5.19.9 Connect and Reconnect Timeslot Escape Services

5.19.9.1 Connect Timeslot Escape Service

OpenScape 4000 supports the ability to connect a special target device (attached to a trunk) to the voice channel(s) of a source device. The Escape service connect timeslot is used to support this non-standard feature.

Precondition for connect timeslot request: source device is in non-idle state, target device is in talk state (this can be achieved by connecting it to a “Help Party” which can be a virtual device).

Connect Timeslot service supports:
- Joint and Separate mode
- Call and talk oriented timeslot connection

5.19.9.1.1 Separate mode

At talk oriented mode the source device always has a partner so the timeslot of the source or the partner device is always available.

For call oriented mode the source device may be connected to a tone (e.g. dial tone). In this case if a connect timeslot service with listen channel is started, then the target device will be connected to the same tone. If the service is started with the talk channel, then the target device will be connected to silence. If the automatic reconnect timeslot mechanism detects that the source device got a talking partner, then the timeslot will be connected as in the talk oriented case.

5.19.9.1.2 Joint mode

If a connect timeslot service with joint mode will be started, then the listen channel and the talk channel of the source device will be mixed using a conference circuit. For joint mode for each source device a conference circuit with two input channels and an output channel will be reserved and used. The output of the conference circuit will be connected to the target device. The input channels of the conference circuit depend on the state of the source device.

5.19.9.1.3 Call and talk oriented Connect Timeslot

If the connect timeslot service has been started, then the state of the source device will be checked. The service can be started if the source device is in a proper state. The proper state depends on the type of the service (call or talk oriented).

Precondition for using the call oriented ConnectTimeslot request is the talk state of the target device and the non-idle state of the source device.

With the talk oriented connect timeslot service it is only possible to connect the timeslot of a source device to a target device, if the source device is in talk state. Trunks are also supported as source devices. Note that joint mode and talk oriented connection are not work together neither for the listen path nor the talk path.

5.19.9.2 Reconnect Timeslot Escape Service

Reconnect Timeslot Escape service stops the connection to the voice channel(s) of the source device.

Index

Symbole

A

B

C

D

E

F

G

R

S

T