ECMA Technical Report TR/68
December 1994

ECMA

Standardizing Information and Communication Systems

Scenarios for Computer Supported Telecommunications Applications (CSTA) Phase II

Phone: +41 22 849.60.00 - **Fax:** +41 22 849.60.01 - **URL:** http://www.ecma.ch - **Internet:** helpdesk@ecma.ch

ECMA Technical Report TR/68
December 1994

ECMA

Standardizing Information and Communication Systems

Scenarios for Computer Supported Telecommunications Applications (CSTA) Phase II

Phone: +41 22 849.60.00 - **Fax:** +41 22 849.60.01 - **URL:** http://www.ecma.ch - **Internet:** helpdesk@ecma.ch

Monica Broxner - TR-068.DOC - 03.05.95 16,09

Brief History

This Technical Report is based upon Phase II of Services for Computer Supported Telecommunications Applications (CSTA) and introduces CSTA Scenarios. This Technical Report uses Standards ECMA-217 ***Services for Computer Supported Telecommunications Applications Phase II*** and ECMA-218 ***Protocol for Computer Supported Telecommunications Applications Phase II*** to illustrate how CSTA services and events may be used in typical call scenarios. It reflects a common understanding of ECMA member companies. Additional phases of this Technical Report are anticipated. This Technical Report is based on the practical experience of ECMA member companies and represents a pragmatic and widely-based consensus.

The Phase II CSTA standards are not fully backwards compatible with the Phase I standards. Although backwards compatibility is an important consideration and has been maintained whenever possible, the addition of new parameters in certain services and events, as well as the deletion of unused Phase I services and the addition of entirely new Phase II services and events, did not allow complete backwards compatibility.

This Technical Report is dedicated to the memory of Terry Wuerfel

This ECMA Technical Report has been adopted by the ECMA General Assembly of December 1994.

Table of contents

	Page
1 Scope	**1**
2 References	**1**
3 Make Call Scenarios	**3**
3.1 Successful Make Call	3
3.2 Successful manual Call (Off-hook dialling)	5
3.3 Make Call meets Called Party Busy	6
3.4 Make Call meets Calling Party Busy	8
3.5 Make Call meets no answer	9
3.6 Make Call to an invalid number	11
3.6.1 Make Call to an invalid number - User already off-hook	12
3.7 Make Call encounters Privilege Violation on Calling Device	13
3.7.1 Make Call encounters Privilege Violation on Calling Device - User off-hook	14
3.7.2 Make Call encounters Privilege Violation on Called Device	15
3.7.3 Make Call encounters Privilege Violation on Called Device - User off-hook	16
3.8 Offnode Make Call (Networked reached)	17
3.9 Make Call while listening to Dial tone	18
3.10 Make Call Hands-free operation	19
3.11 Successful manual Offnode Call ("Overlap" off-hook dialling)	20
4 Successful ISDN ("En-bloc") Call	**21**
5 Call Release Scenarios	**23**
5.1 Call Release	23
5.2 The CSTA Blocked Call State	24
6 Consultation Calls	**25**
6.1 Successful Consultation Call	25
6.2 Consultation Call Rejected	27
6.3 Consulted party busy	28
6.4 Consulted party clears - consulting party receives dial tone	30
6.5 Held party clears	31
7 Successful Alternate Call	**32**
8 Transfers	**33**
8.1 Successful Supervised Transfer	33
8.2 Successful Unsupervised Transfer	34
8.3 Single Step Transfer	35

8.4 Transfer on Busy 37

9 Conference Calls 39
9.1 Conference Call Service 39
9.2 Multiple Party (Meet Me) Conference 40
9.3 Single Step Conference 42

10 Divert Deflection Service 43

11 Call Completion Service 44

12 A Manually Invoked Call Park 46

13 Route Services 47
13.1 Route Request Service 47
13.2 Route Used Service 49
13.3 Re-Route Service 51

14 Incoming Calls 53
14.1 Successful incoming Call 53
14.2 Incoming call to ACD with no available agents 55

1 Scope

Services for Computer Supported Telecommunications Applications are defined by Standard ECMA-217 and the Protocol for those services by Standard ECMA-218. Our intention here is to illustrate how, for a range of call scenarios, the services offered by ECMA CSTA are intended to be used by application developers and switch manufacturers.

Each scenario includes a textual description and an illustration. Illustrations use the same key as described within ECMA-217 section 9.3. For each scenario, message sequences are listed for all monitored devices - call type monitors have not been illustrated. All devices have device type monitors set with no events masked. The *Activity* column includes a brief description of the event or service invocation, the *Monitored Device* column lists events and service invocations with the associated parameter list. Optional parameters within the *Monitored Device* column are shown in italics. DeviceIDs are illustrated by D*n* and ConnectionIDs in the form D*n*C*n*. 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*.

CSTA provides an event reporting service which contains three parameters. These parameters, crossRefIdentifier, eventSpecificInfo and extensions are fully defined by ECMA-218. Within these scenarios only the content of the eventSpecificInfo parameter within the CSTA Event Reporting Service is shown.

These scenarios show a strict order of messaging (i.e. a service request is always followed by a service acknowledgement). However, in a true implementation an acknowledgement to a service request may be received in an asynchronous mode by the computing function. Events are generated throughout the life of a call and as such represent in these scenarios the passing of time.

The precise moment in relation to the switching function activity at which a response is generated, shall be implementation- and Service- dependent. For example, for the Hold Call Service some implementations may generate the Service Response after validating the correctness of the request and when they initiate the requested Service. Other implementations may delay the response until the Hold has completed (or is guaranteed to complete). In this case, a failure of the requested switching Service is reflected in the Service Response.

The scenarios are only for information and as such ECMA-217 (Services) and ECMA-218 (Protocol) standards may define additional options or parameters. The purpose of this Technical Report is to provide examples of some CSTA Service invocations and illustrate associated call event reports. It is not an exhaustive document and some implementations may not perform as illustrated within this document. A method of providing these scenarios as a collection of smaller building blocks which may then be connected together to form more complex scenarios is under study.

SubjectDeviceID is used in some event reports to specify which device the report refers to. If the SubjectDeviceID has had a monitor invoked upon it then this data is not required and so the implicit NULL encoding for notRequired should be returned, as defined by ECMA-218. In these scenarios, as all devices have active device monitors, the SubjectDeviceID is shown as /NR to indicate Not Required. NK is used to indicate notKnown where appropriate. Within these Scenarios, ( ) indicates that the content is not specified by these scenarios and that anything defined within the protocol is valid.

2 References

ECMA TR/52	Computer Supported Telecommunications Applications (CSTA) (1990)
ECMA-217	Services for Computer Supported Telecommunications Applications (CSTA) Phase II (1994)
ECMA-218	Protocol for Computer Supported Telecommunications Applications (CSTA) Phase II (1994)

3 Make Call Scenarios

This section includes examples of successful Make Calls, initiated manually and by applications, Make Calls by and to busy parties and calls which do not get answered. There are four examples of calls rejected because of privilege violations. Calls made to devices outside of the CSTA domain and an ISDN en-bloc call are illustrated. A Make Call Response, positive or negative, may indicate the result of the Make Call Request. For some switch implementations the result of the request is determined from the CSTA events which follow the Make Call Response.

3.1 Successful Make Call

This scenario initiates a call from device D1 to device D2. In this scenario both devices are free and valid, device D1 is permitted to make the call and the call reaches establishment.

BEFORE AFTER

Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3	Comments
A MakeCall to a valid device is invoked on behalf of device D1.	MakeCallRequest • callingDevice • calledDirectoryNumber • *deviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1 D2 *( )* *( )* *( )* *( )* *( )*				
Acknowledgement.	MakeCallResult • initiatedCall • *extensions*	 D1C1 *( )*				
Indication that service has been initiated from this device.	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C1 *Initiated* *makeCall*				*The generation of this event is switch-specific.*
Device D1 lifts handset ready for call.	OriginatedEvent • originatedConnection • callingDevice • calledDevice • *originatingDevice* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR D2 *( )* *Connected* *( )* *newCall*				
Device D2 begins to ring and D1 listens to ringing tone.	DeliveredEvent • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2 D1 D2 NR *( )* *Connected* *( )* *newCall*	DeliveredEvent • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2/NR D1 D2 NR *( )* *Alerting* *( )* *newCall*		

Device D2 answers the call.	EstablishedEvent • establishedConnection • answeringDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2 D1 D2 NR *( )* *Connected* *( )* *newCall*	EstablishedEvent • establishedConnection • answeringDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2/NR D1 D2 NR *( )* *Connected* *( )* *newCall*		*Device D2 manually answers the call by lifting handset.*

3.2 Successful manual Call (Off-hook dialling)

In this scenario the instrument associated with device D1 is manually lifted to initiate a call from device D1 to device D2. Both devices are initially free and valid, device D1 is permitted to make the call and the call reaches establishment.

BEFORE AFTER

Activity	MONITORED DEVICE D1	MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
Device D1 goes off-hook and receives dial tone.	ServiceInitiatedEvent • initiatedConnection D1C1 • *localConnectionInfo* *Initiated* • *cause* *newCall*			*Device is authorised to make calls. This event is always generated for a manual MakeCall.*
Dialling completed - call set-up taking place.	OriginatedEvent • originatedConnection D1C1 • callingDevice D1/NR • calledDevice D2 • *originatingDevice* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*			
Device D2 is free and begins to ring.	DeliveredEvent • connection D2C1 • alertingDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*	DeliveredEvent • connection D2C1 • alertingDevice D2/NR • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Alerting* • *correlatorData* *( )* • *cause* *newCall*		*Assuming that no diverts are in operation.*
AnswerCall service is invoked on behalf of device D2.		AnswerCallRequest • callToBeAnswered D2C1 • *extensions* *( )*		*An error will be sent if the device is not able to answer the call without manual intervention.*
Acknowledgement.		AnswerCallResult • *extensions* *( )*		
Device D2 answers the call.	EstablishedEvent • establishedConnection D2C1 • answeringDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*	EstablishedEvent • establishedConnection D2C1 • answeringDevice D2/NR • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*		

3.3 Make Call meets Called Party Busy

The application initiates a call from device D1 to device D2. Device D1 is prompted to lift the handset. Device D2 is busy and has not invoked a divert. The call fails because the called party is busy. Device D1 clears without invoking a supplementary service. In this scenario the fact that device D2 is busy is not indicated in the MakeCallResult but by a subsequence of event reports.

BEFORE AFTER

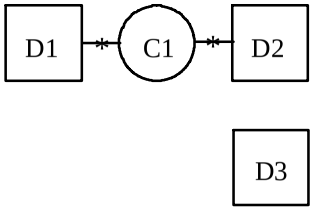
Activity	MONITORED DEVICE D1	MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
MakeCall service is invoked on behalf of device D1.	MakeCallRequest • callingDevice D1 • calledDirectoryNumber D2 • *deviceProfile* *( )* • *accountCode* *( )* • *authCode* *( )* • *correlatorData* *( )* • *extensions* *( )*			
Acknowledgement.	MakeCallResult • initiatedCall D1C1 • *extensions* *( )*			
Device D1 notified of initiating call.	ServiceInitiatedEvent • initiatedConnection D1C1 • *localConnectionInfo* *Initiated* • *cause* *makeCall*			*The generation of this event is switch-specific.*
Device D1 lifts handset and establishes the calling connection of the call.	OriginatedEvent • originatedConnection D1C1 • callingDevice D1/NR • calledDevice D2 • *originatingDevice* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*			
Device D2 is busy - the call cannot be completed. Device D1 hears busy tone.	FailedEvent • failedConnection D2C1 • failingDevice D2 • calledDevice D2 • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *Busy*	FailedEvent • failedConnection D2C1 • failingDevice D2/NR • calledDevice D2 • *localConnectionInfo* *Failed* • *correlatorData* *( )* • *cause* *Busy*		
Device D1 now replaces handset.	ConnectionClearedEvent • droppedConnection D1C1 • releasingDevice D1/NR • *localConnectionInfo* *Null* • *correlatorData* *( )* • *cause* *normal Clearing*	ConnectionClearedEvent • droppedConnection D1C1 • releasingDevice D1 • *localConnectionInfo* *Failed* • *correlatorData* *( )* • *cause* *normal Clearing*		

Failed connection D2C1 also clears.		ConnectionClearedEvent • droppedConnection D2C1 • releasingDevice D2 • *localConnectionInfo Null* • *correlatorData ( )* • *cause normal Clearing*		*Failed connections must also clear and therefore report clear connection events.*

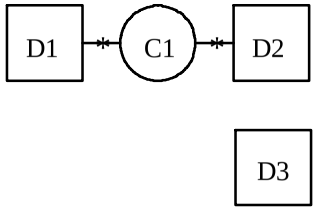
3.4 Make Call meets Calling Party Busy

CSTA Make Call fails because the calling party, device D1, is busy on a call to device D2 when a MakeCallRequest is issued on behalf of device D1. On certain types of device the switch may be able to launch the call on a separate line appearance. This operation is implementation-dependent and is therefore not illustrated here.

BEFORE



AFTER



Activity	MONITORED DEVICE D1		MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
User initiates Make Call.	MakeCallRequest • callingDevice • calledDirectoryNumber • *deviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1 D3 *( )* *( )* *( )* *( )* *( )*			
Make Call fails because calling party is busy.	MakeCallError • stateErrors	 invalid object state			

3.5 Make Call meets no answer

The application initiates a call from device D1 to device D2 and prompts the user of device D1 to lift the handset. Device D1 and device D2 are free but device D2 doesn't answer the call. Device D1 replaces the handset to clear the call.

BEFORE AFTER

Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3	Comments
User initiates Make Call.	MakeCallRequest • callingDevice • calledDirectoryNumber • *deviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1 D2 *( )* *( )* *( )* *( )* *( )*				
Acknowledgement.	MakeCallResult • initiatedCall • *extensions*	 D1C1 *( )*				
Device D1 notified of initiating call.	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C1 *Initiated* *makeCall*				*The generation of this event is switch-specific.*
Device D1 lifts handset to originate calling end connection.	OriginatedEvent • originatedConnection • callingDevice • calledDevice • *originatingDevice* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR D2 *( )* *Connected* *( )* *newCall*				
Device D2 is free and begins to alert.	DeliveredEvent • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2 D1 D2 NR *( )* *Connected* *( )* *newCall*	DeliveredEvent • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2/NR D1 D2 NR *( )* *Alerting* *( )* *newCall*		
D1 replaces handset because of failure to connect.	ConnectionClearedEvent • droppedConnection • releasingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR *Null* *( )* *normal Clearing*	ConnectionClearedEvent • droppedConnection • releasingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1 *Alerting* *( )* *normal Clearing*		

Connection D2C1 clears as a result of D1 clearing.		ConnectionClearedEvent • droppedConnection • releasingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2/NR *Null* *( )* *normal Clearing*	*The cause value "callNotAnswered" only applies if a call is cleared by a switch as a result of a timer rather than by the caller replacing their handset.*

3.6 Make Call to an invalid number

This scenario initiates a call from device D1 to device D2. In this scenario device D1 is free, valid and permitted to make the call. Device D2 is actually an invalid number (not part of any known numbering plan) and the call fails.

BEFORE

AFTER

Activity	MONITORED DEVICE D1		MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
MakeCall service to an invalid number is initiated on behalf of device D1.	MakeCallRequest • callingDevice • calledDirectoryNumber • *deviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1 D2 *( )* *( )* *( )* *( )* *( )*			
Acknowledgement.	UniversalFailure • operationalErrors	 invalid Called Device			

3.6.1 Make Call to an invalid number - User already off-hook

This scenario invokes a call that was already initiated by a user going off-hook on a telephone. The call is from device D1 to D2. In this scenario device D1 is free, valid and permitted to make the call. Device D2 is actually an invalid number (not part of any known numbering plan) and the call fails.

BEFORE AFTER

Activity	MONITORED DEVICE D1		MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
Device D1 manually initiates a call by going off-hook.	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C1 *Initiated* *newCall*			
MakeCall service to an invalid device invoked on behalf of device D1.	MakeCallRequest • callingDevice • calledDirectoryNumber • *deviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1 D2 *( )* *( )* *( )* *( )* *( )*			
Acknowledgement.	UniversalFailure • operationalErrors	 invalid Called Device			*D2 is a number that is unknown to the switching function.*
Device D1 gets a "failed" tone.	FailedEvent • failedConnection • failingDevice • calledDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR D2 *Failed* *( )* *destNot Obtainable*			*D2 is a number that is unknown to the switching function.*
Connection cleared as D1 replaces handset.	ConnectionClearedEvent • droppedConnection • releasingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR *Null* *( )* *normal Clearing*			

3.7 Make Call encounters Privilege Violation on Calling Device

This scenario initiates a call from device D1 to device D2. In this case device D1 is not permitted to attempt calls resulting in a privilege violation and the call fails.

BEFORE AFTER

Activity	MONITORED DEVICE D1		MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
MakeCall service to device D2 invoked on behalf of device D1.	MakeCallRequest • callingDevice • calledDirectoryNumber • *deviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1 D2 *( )* *( )* *( )* *( )* *( )*			
Acknowledgement.	UniversalFailure • operationalErrors	 Privilege ViolationOn Calling Device			

3.7.1 Make Call encounters Privilege Violation on Calling Device - User off-hook

This scenario dials a call that was already initiated by a user going off-hook on a telephone. The call is from device D1 to device D2. In this scenario device D1 is not permitted to make calls - so calling device D2 causes a privilege violation, at device D1, and the call fails. Note that off-hook in itself may not be a privilege violation as it is also a way to activate features.

BEFORE AFTER

Activity	MONITORED DEVICE D1		MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
Device D1 manually initiates a call by going off-hook.	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C1 *Initiated* *newCall*			
MakeCall service to an invalid device is invoked on behalf of device D1.	MakeCallRequest • callingDevice • calledDirectoryNumber • *deviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1 D2 *( )* *( )* *( )* *( )* *( )*			
Acknowledgement.	UniversalFailure • operationalErrors	 privilege ViolationOn Calling Device			*Outbound calling is not allowed from this device.*
Device D1 gets a "failed" tone.	FailedEvent • failedConnection • failingDevice • calledDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR D2 *Failed* *( )* *destNot Obtainable*			*D2 is a number that is not reachable from device D1.*
Connection cleared as D1 replaces handset.	ConnectionClearedEvent • droppedConnection • releasingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR *Null* *( )* *normal Clearing*			*Failed connections must also be cleared.*

3.7.2 Make Call encounters Privilege Violation on Called Device

This scenario initiates a call from device D1 to device D2. In this scenario device D1 is free, valid and permitted to attempt calls - but by calling device D2, D1 actually causes a privilege violation and the call fails.

BEFORE AFTER

Activity	MONITORED DEVICE D1		MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
Device D1 initiates a MakeCall to an invalid device.	MakeCallRequest • callingDevice • calledDirectoryNumber • *deviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1 D2 *( )* *( )* *( )* *( )* *( )*			
Acknowledgement.	UniversalFailure • operationalErrors	 privilege ViolationOn Called Device			*Calling D2 is not allowed for this device.*

3.7.3 Make Call encounters Privilege Violation on Called Device - User off-hook

This scenario dials a call that was already initiated by a user going off-hook on a telephone. The call is from device D1 to device D2. In this scenario device D1 is free, valid and permitted to make calls - but by calling device D2, D1 causes a privilege violation and the call fails.

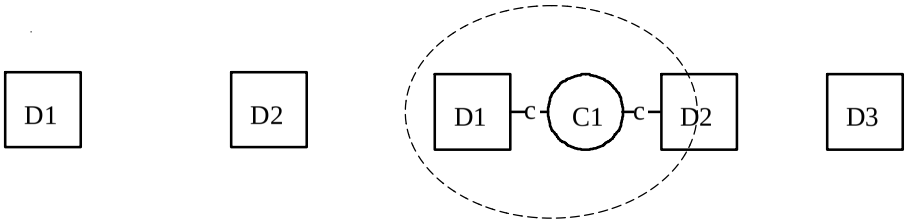
BEFORE AFTER

Activity	MONITORED DEVICE D1		MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
Device D1 manually initiates a call by going off-hook.	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C1 *Initiated* *newCall*			
MakeCall service to an invalid device is invoked on behalf of device D1.	MakeCallRequest • callingDevice • calledDirectoryNumber • *deviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1 D2 *( )* *( )* *( )* *( )* *( )*			
Acknowledgement.	UniversalFailure • operationalErrors	 privilege ViolationOn Called Device			*Calling D2 is not allowed for this device.*
Device D1 gets a "failed" tone.	FailedEvent • failedConnection • failingDevice • calledDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR D2 *Failed* *( )* *destNot Obtainable*			*D2 is a number that is not reachable from device D1.*
Connection cleared as D1 replaces handset.	ConnectionClearedEvent • droppedConnection • releasingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR *Null* *( )* *normal Clearing*			

3.8 Offnode Make Call (Networked reached)

A CSTA make call is invoked and the call is routed out of the CSTA sub-domain. Device D1 is prompted to lift the handset. D3, which lies outside the CSTA domain, can not be monitored and therefore no events will be seen for that device. Event information after the NetworkReached event is not available.

BEFORE AFTER



Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3	Comments
User initiates a call.	MakeCallRequest • callingDevice • calledDirectoryNumber • *deviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1 D3 *( )* *( )* *( )* *( )* *( )*				
Acknowledgement.	MakeCallResult • initiatedCall • *extensions*	 D1C1 *( )*				
Device D1 notified of initiating call.	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C1 *Initiated* *makeCall*				*The generation of this event is switch-specific.*
Device D1 lifts handset ready for call.	OriginatedEvent • originatedConnection • callingDevice • calledDevice • *originatingDevice* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR D3 *D2* *Connected* *( )* *newCall*				
The call leaves the CSTA domain.	NetworkReachedEvent • outboundConnection • trunkUsed • calledDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2 D3 *Connected* *( )* *newCall*	NetworkReachedEvent • outboundConnection • trunkUsed • calledDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2/NR D3 *Connected* *( )* *newCall*		*The Called Device information will be complete. Within a networking environment the trunk may not be visible.*

3.9 Make Call while listening to Dial tone

This scenario originates a call from device D1 to device D2. In this scenario device D1 is free, valid and permitted to make the call - and device D2 is a valid number. The switching function dials the call for which the user of device D1 has already gone off-hook.

BEFORE AFTER

Activity	MONITORED DEVICE D1		MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
Device D1 goes off-hook.	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C1 *Initiated* *newCall*			
MakeCall service to device D2 invoked on behalf of device D1.	MakeCallRequest • callingDevice • calledDirectoryNumber • *deviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1 D2 *( )* *( )* *( )* *( )* *( )*			
Acknowledgement.	MakeCallResult • initiatedCall • *extensions*	 D1C1 *( )*			*Call now proceeds.*
Call proceeds from device D1.	OriginatedEvent • originatedConnection • callingDevice • calledDevice • *originatingDevice* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR D2 *( )* *Connected* *( )* *newCall*			
	...sequence proceeds as other Make Call scenarios...				

3.10 Make Call Hands-free operation

This scenario originates a call from device D1 to device D2. In this scenario device D1 is free, valid and permitted to make the call - and device D2 is a valid number. The switching function dials the call and automatically forces device D1 to go off-hook. Switching functions often have the capability to force "digital" sets to go off-hook without the user having to participate. With "analogue" sets it is necessary for the switching function to prompt the user to lift the handset.

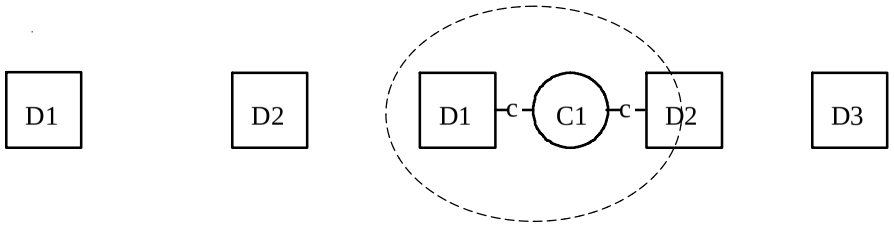
BEFORE AFTER

Activity	MONITORED DEVICE D1		MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
MakeCall to device D2 invoked on behalf of device D1.	MakeCallRequest • callingDevice • calledDirectoryNumber • *deviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1 D2 *( )* *( )* *( )* *( )* *( )*			
Acknowledgement.	MakeCallResult • initiatedCall • *extensions*	 D1C1 *( )*			*Call now proceeds.*
Call proceeds from device D1. The switching function automatically forced D1 off-hook without the user participating.	OriginatedEvent • originatedConnection • callingDevice • calledDevice • *originatingDevice* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR D2 *( )* *Connected* *( )* *newCall*			
	....scenario proceeds as for other Make Call scenarios...				

3.11 Successful manual Offnode Call ("Overlap" off-hook dialling)

In this scenario the instrument associated with device D1 is manually lifted to initiate a call, and the call is routed out of the CSTA sub-domain. The device dials a trunk or a trunk group prefix first and then the external number. Device D3 is not within the CSTA domain and therefore cannot be monitored.

BEFORE AFTER

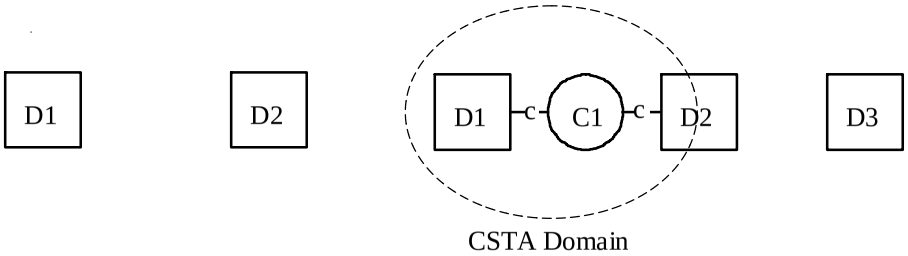


Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3	Comments
Device D1 goes off-hook and receives dial tone.	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C1 *Initiated* newCall				
Device D1 completes dialling the trunk access. The call leaves the CSTA domain.	NetworkReachedEvent • outboundConnection • trunkUsed • calledDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2 NK *Initiated* *( )* *newCall*	NetworkReachedEvent • outboundConnection • trunkUsed • calledDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2/NR NK *Connected* *( )* *newCall*		*The Called Device information will be complete only when D1 completes its dialling. Within a networking environment the trunk may not be visible.*
Device D1 completes dialling.	OriginatedEvent • originatedConnection • callingDevice • calledDevice • *originatingDevice* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR D3 *D2* *Connected* *( )* *newCall*	OriginatedEvent • originatedConnection • callingDevice • calledDevice • *originatingDevice* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1 D3 *D2* *Connected* *( )* *newCall*		
Device D1 receives an answer from D3 or the switch assumes an answer based upon a timer.	EstablishedEvent • establishedConnection • answeringDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D3 D1 D3 NR *D2C1* *Connected* *( )* *network Signal*	EstablishedEvent • establishedConnection • answeringDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D3 D1 D3 NR *D2C1* *Connected* *( )* *network Signal*		*Depending on the type of signalling employed over the trunk a delivered event may be seen.*

4 Successful ISDN ("En-bloc") Call

In this scenario the instrument associated with device D1 manually initiates a call, and the call is routed out of the CSTA sub-domain via an ISDN trunk.

BEFORE AFTER



Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3	Comments
Device D1 goes off-hook and receives dial tone.	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C1 *Initiated* *newCall*				
Device D1 completes dialling.	OriginatedEvent • originatedConnection • callingDevice • calledDevice • *originatingDevice* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR D3 *D2* *Connected* *( )* *newCall*				
The call leaves the CSTA domain.	NetworkReachedEvent • outboundConnection • trunkUsed • calledDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2 D3 *Connected* *( )* *newCall*	NetworkReachedEvent • outboundConnection • trunkUsed • calledDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2/NR D3 *Connected* *( )* *newCall*		
Device D3 is being alerted and D1 listens to ringing tone.	DeliveredEvent • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D3 D1 D3 NR *D2C1* *Connected* *( )* *network Signal*	DeliveredEvent • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D3 D1 D3 NR *D2C1* *Connected* *( )* *network Signal*		*Corresponds to the Alert message in ISDN. Device D2 acts as a proxy device for device D3.*

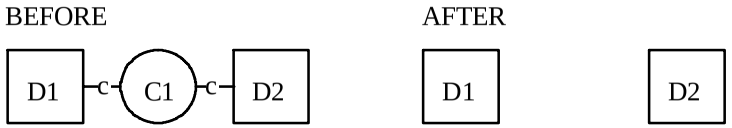
Device D3 answers.	EstablishedEvent • establishedConnection • answeringDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D3 D1 D3 NR *D2C1* *Connected* *( )* *network Signal*	EstablishedEvent • establishedConnection • answeringDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D3 D1 D3 NR *D2C1* *Connected* *( )* *network Signal*		*Corresponds to the Answer message in ISDN.*

5 Call Release Scenarios

This section includes examples of calling and called parties releasing. The latter is an example of how a call can reside in the CSTA blocked call state.

5.1 Call Release

Device D1 and D2 are in an established call. Device D1 decides to clear the call - a local end disconnect.

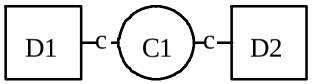


Activity	MONITORED DEVICE D1	MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
Clear Connection service invoked on behalf of device D1.	ClearConnectionRequest • connectionToBeCleared D1C1 • *extensions* *( )*			
Acknowledgement.	ClearConnectionResult • *extensions* *( )*			
Connection D1C1 is cleared.	ConnectionClearedEvent • droppedConnection D1C1 • releasingDevice D1/NR • *localConnectionInfo* *Null* • *correlatorData* *( )* • *cause* *normal Clearing*	ConnectionClearedEvent • droppedConnection D1C1 • releasingDevice D1 • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *normal Clearing*		*A connection cleared event does not necessarily mean that a handset has been replaced. An application will determine from the connection cleared event who initiated clearing.*
Connection D2C1 cleared as a result of D1C1 clearing.		ConnectionClearedEvent • droppedConnection D2C1 • releasingDevice D2/NR • *localConnectionInfo* *Null* • *correlatorData* *( )* • *cause* *normal Clearing*		

5.2 The CSTA Blocked Call State

Device D1 and D2 are in an established call. Device D2 decides to clear the call - a far end disconnect which places the call in the blocked state.

BEFORE



AFTER

Activity	MONITORED DEVICE D1	MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
Clear Connection service invoked on behalf of device D2.		ClearConnectionRequest • connectionToBeCleared D2C1 • *extensions* *( )*		*When a call consists of only two devices a ClearConnection is equivalent to ClearCall.*
Acknowledgement.		ClearConnectionResult • *extensions* *( )*		
Device D2 hangs up first.	ConnectionClearedEvent • droppedConnection D2C1 • releasingDevice D2 • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *normal Clearing*	ConnectionClearedEvent • droppedConnection D2C1 • releasingDevice D2/NR • *localConnectionInfo* *Null* • *correlatorData* *( )* • *cause* *normal Clearing*		*Device D1 is still off-hook.*
Connection D1C1 moves into the failed state.	FailedEvent • failedConnection D1C1 • failingDevice D1/NR • calledDevice D2 • *localConnectionInfo* *Failed* • *correlatorData* *( )* • *cause* *blocked*			*The CSTA Simple Call state Blocked applies when the local connection, D1C1, is in the Failed state and the other connection, D2C1, is in the Null state.*
Device D1 hangs up.	ConnectionClearedEvent • droppedConnection D1C1 • releasingDevice D1/NR • *localConnectionInfo* *Null* • *correlatorData* *( )* • *cause* *normal Clearing*			

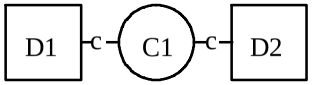
6 Consultation Calls

This section includes examples of invoking a consultation call which is rejected, making a consultation call where the consulted party clears, making a consultation call where the held party clears, and invoking enquiry calls to non-existent devices.

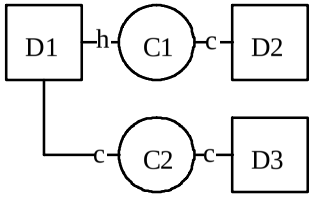
6.1 Successful Consultation Call

A call between device D1 and device D2 is already connected. Device D1 decides to invoke a consultation call to another device, D3. This call is successful.

BEFORE



AFTER



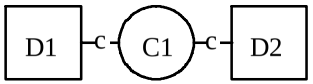
Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3	Comments
Consultation Call service to device D3 is invoked on behalf of device D1.	ConsultationCallRequest • existingCall • consultedDevice • *consultedDeviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1C1 D3 *( )* *( )* *( )* *( )* *( )*				
Acknowledgement.	ConsultationCallResult • initiatedCall • *extensions*	 D1C2 *( )*				
Connection placed on hold.	HeldEvent • heldConnection • holdingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR *Hold* *( )* *consultation*	HeldEvent • heldConnection • holdingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1 *Connected* *( )* *consultation*		*In some implementations the cause consultation may not be supported.*
Application notified of initiating call.	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C2 *Initiated* *newCall*				*The generation of this event is switch-specific.*
The Consultation Call is launched.	OriginatedEvent • originatedConnection • callingDevice • calledDevice • *originatingDevice* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C2 D1/NR D3 *D2* *Connected* *( )* *newCall*				*At this point a consultation call may follow any of the MakeCall scenarios produced in this Technical Report.*

Device D3 begins to alert.	DeliveredEvent • connection D3C2 • alertingDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*		DeliveredEvent • connection D3C2 • alertingDevice D3/NR • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Alerting* • *correlatorData* *( )* • *cause* *newCall*	
New call established.	EstablishedEvent • establishedConnection D3C2 • answeringDevice D3 • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*		EstablishedEvent • establishedConnection D3C2 • answeringDevice D3/NR • callingDevice D1 • calledDevice D3 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*	*Assuming that the called device actually answers the call.*

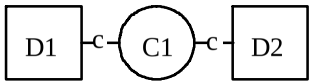
6.2 Consultation Call Rejected

Device D1, which already has an active call, invokes the Consultation Call service which provides the compound action of the Hold Call service and the Make Call service. The Hold Call request fails because a hold is not allowed. Reasons for not allowing hold include calls which involve an operator where operators may not be placed on hold, or a network hold request which is rejected.

BEFORE



AFTER

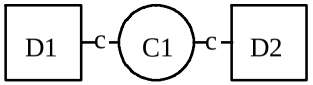


Activity	MONITORED DEVICE D1		MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
Application issues Consultation Call which invokes a hold call request.	ConsultationCallRequest • existingCall • calledDirectoryNumber • *calledDeviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1C1 D3 *( )* *( )* *( )* *( )* *( )*			
Hold request rejected.	ConsultationCallError • operationalError	 invalid Feature			*Example - an operator may not be placed on hold.*

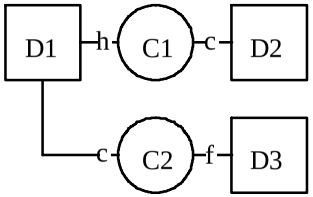
6.3 Consulted party busy

Device D1 is active in a call with device D2 and decides to invoke a consultation call to another device. The consultation call fails because the specified destination device is busy. Device D1 clears the enquiry call.

BEFORE



AFTER



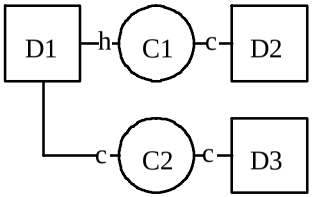
Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3	Comments
Consultation Call service to device D3 is invoked on behalf of device D1.	ConsultationCallRequest • existingCall • calledDirectoryNumber • *calledDeviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1C1 D3 *( )* *( )* *( )* *( )* *( )*				
Acknowledgement.	ConsultationCallResult • initiatedCall • *extensions*	 D1C2 *( )*				
Call C1 is put on hold.	HeldEvent • heldConnection • holdingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR *Hold* *( )* *consultation*	HeldEvent • heldConnection • holdingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1 *Connected* *( )* *consultation*		*In some implementations the cause consultation may not be supported.*
Application notified of initiating call.	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C2 *Initiated* *newCall*				
Device D1 completes dialling the consultation party number.	OriginatedEvent • originatingDevice • originatedConnection • callingDevice • calledDevice • *originatingDevice* • *localConnectionInfo* • *correlatorData* • *cause*	 D1/NR D1C2 D1/NR D3 *D2* *Connected* *( )* *newCall*				*At this point a consultation call may follow any of the MakeCall scenarios produced in this Technical Report.*

The connection to device D3 fails due to busy.	FailedEvent • failedConnection D3C2 • failingDevice D3 • calledDevice D3 • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *Busy*		FailedEvent • failedConnection D3C2 • failingDevice D3/NR • calledDevice D3 • *localConnectionInfo* *Failed* • *correlatorData* *( )* • *cause* *Busy*	
Originator clears call C2 and does not invoke the call completion service.	ConnectionClearedEvent • droppedConnection D1C2 • releasingDevice D1/NR • *localConnectionInfo* *Null* • *correlatorData* *( )* • *cause* *normal Clearing*		ConnectionClearedEvent • droppedConnection D1C2 • releasingDevice D1 • *localConnectionInfo* *Failed* • *correlatorData* *( )* • *cause* *normal Clearing*	
Failed connection D3C2 clears.			ConnectionClearedEvent • droppedConnection D3C2 • releasingDevice D3 • *localConnectionInfo* *Null* • *correlatorData* *( )* • *cause* *normal Clearing*	*Device D1 may now go on and reconnect to the call on hold. This may be automatic.*

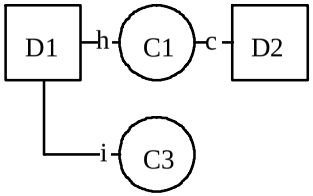
6.4 Consulted party clears - consulting party receives dial tone

Device D1 is active in a call with device D2 and decides to make a Consultation call to another device. The consultation call is successful and when complete the consulted party clears. Device D1 is then provided with dial tone to enable them to either invoke the retrieve call service to reconnect to the held party or make another consultation call.

BEFORE



AFTER

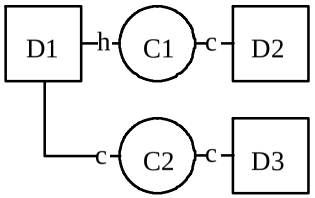


Activity	MONITORED DEVICE D1	MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
The Consulted party clears the call using the Clear Connection Service.			ClearConnectionRequest • connectionToBeCleared D3C2 • *extensions* *( )*	
Acknowledgement.			ClearConnectionResult • *extensions* *( )*	
Connection D3C2 clears.	ConnectionClearedEvent • droppedConnection D3C2 • releasingDevice D3 • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *normal Clearing*		ConnectionClearedEvent • droppedConnection D3C2 • releasingDevice D3/NR • *localConnectionInfo* *Null* • *correlatorData* *( )* • *cause* *normal Clearing*	
Switch clears connection D1C2.	ConnectionClearedEvent • droppedConnection D1C2 • releasingDevice D1/NR • *localConnectionInfo* *Null* • *correlatorData* *( )* • *cause* *normal Clearing*			
The switch provides dial tone for a new call or feature invocation - call C3.	ServiceInitiatedEvent • initiatedConnection D1C3 • *localConnectionInfo* *Initiated* • *cause* *newCall*			

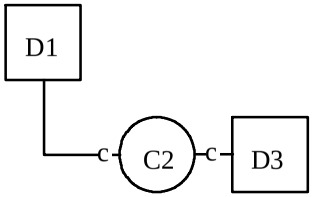
6.5 Held party clears

Device D1 is active in a Consultation call with device D3; the connection to device D2 is on hold. During the Consultation call device D2, the held party, decides to clear. The held call and associated resources are cleared while the active call continues uninterrupted.

BEFORE



AFTER

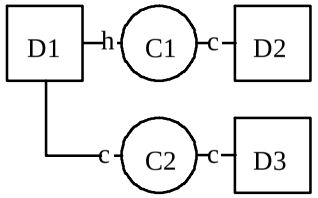


Activity	MONITORED DEVICE D1	MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
Clear Connection service invoked on behalf of device D2.		ClearConnectionRequest • connectionToBeCleared D2C1 • *extensions* *( )*		
Acknowledgement.		ClearConnectionResult • *extensions* *( )*		
Connection D2C1 clears.	ConnectionClearedEvent • droppedConnection D2C1 • releasingDevice D2 • *localConnectionInfo* *Hold* • *correlatorData* *( )* • *cause* *normal Clearing*	ConnectionClearedEvent • droppedConnection D2C1 • releasingDevice D2/NR • *localConnectionInfo* *Null* • *correlatorData* *( )* • *cause* *normal Clearing*		
Automatic clearing of connection D1C1.	ConnectionClearedEvent • droppedConnection D1C1 • releasingDevice D1/NR • *localConnectionInfo* *Null* • *correlatorData* *( )* • *cause* *normal Clearing*			*The Call between devices D1 and D3 continues uninterrupted.*

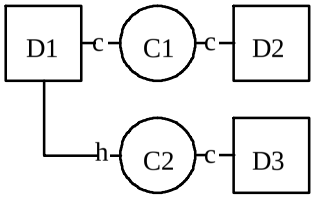
7 Successful Alternate Call

The Alternate Call service provides the compound action of the Hold Call service followed by the Retrieve Call service. An existing active call is placed on hold and a previously held call is re-connected as illustrated by this scenario.

BEFORE



AFTER



Activity	MONITORED DEVICE D1	MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
Alternate Call service is invoked on behalf of device D1.	AlternateCallRequest • heldCall D1C1 • activeCall D1C2 • *extensions* *( )*			
Acknowledgement.	AlternateCallResult • *extensions* *( )*			
Connection to D3 is put on hold.	HeldEvent • heldConnection D1C2 • holdingDevice D1/NR • *localConnectionInfo* *Hold* • *correlatorData* *( )* • *cause* *Alternate*		HeldEvent • heldConnection D1C2 • holdingDevice D1 • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *Alternate*	*In some implementations the cause value "Alternate" may not be supported.*
Switch acknowledges retrieval of the held call to device D2.	RetrievedEvent • retrievedConnection D1C1 • retrievingDevice D1/NR • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *Alternate*	RetrievedEvent • retrievedConnection D1C1 • retrievingDevice D1 • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *Alternate*		

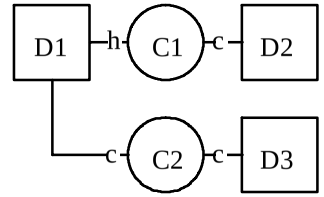
8 Transfers

This section includes examples of a supervised and unsupervised transfer and a single step transfer.

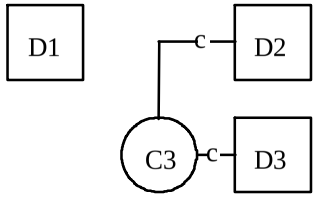
8.1 Successful Supervised Transfer

Device D1 has a call to device D3 connected and a call to device D2 which is on hold. The Transfer Call service provides the ability to connect devices D2 and D3. Device D1 drops out of the call. This is a supervised transfer because device D1 established the call to device D3 before transferring the call.

BEFORE



AFTER

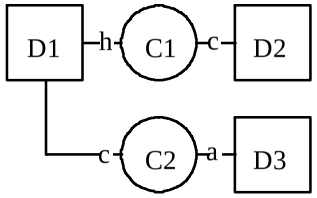


Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3		Comments
Transfer Call service is invoked on behalf of device D1.	TransferCallRequest • heldCall • activeCall • *extensions*	 D1C1 D1C2 *( )*					
Acknowledgement.	TransferCallResult • *transferredCall* • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *extensions*	 *D3C3* *D2C3* *D2* *D2C1* *D3C3* *D3* *D3C2* *( )*					
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.	TransferredEvent • primaryOldCall • secondaryOldCall • transferringDevice • transferredDevice • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1C2 D1/NR D3 *D2C3* *D2* *D2C1* *D3C3* *D3* *D3C2* *Null* *( )* *Transfer*	TransferredEvent • primaryOldCall • *secondaryOldCall* • transferringDevice • transferredDevice • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 *( )* D1 D3 D2C3 D2 D2C1 D3C3 D3 D3C2 *Connected* *( )* *Transfer*	TransferredEvent • primaryOldCall • *secondaryOldCall* • transferringDevice • transferredDevice • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D3C2 *( )* D1 D3/NR D2C3 D2 D2C1 D3C3 D3 D3C2 *Connected* *( )* *Transfer*	*The result is that connections D2C1, D1C1, D1C2 and D3C2 are cleared. SecondaryOldCall parameter is only included if events have been reported for the connection identifier.*

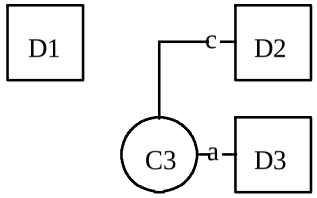
8.2 Successful Unsupervised Transfer

Device D1 has a call to device D3 which is alerting and a call to device D2 which is on hold. The Transfer Call service provides the ability to connect devices D2 and D3. Device D1 drops out of the call. This is an unsupervised transfer because device D1 transferred the call to device D3 before completing the establishment.

BEFORE



AFTER

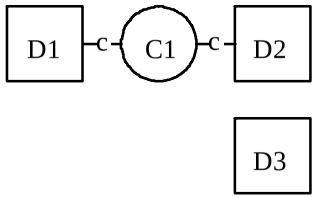


Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3		Comments
Transfer Call service is invoked on behalf of device D1.	TransferCallRequest • heldCall • activeCall • *extensions*	 D1C1 D1C2 *( )*					
Transfer call is accepted.	TransferCallResult • *transferredCall* • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *extensions*	 *D3C3* *D2C3* *D2* *D2C1* *D3C3* *D3* *D3C2* *( )*					
Calls between D1,D2 and D1,D3 are released. Connections between D2,D1 and D3,D1 are replaced with a single connection between D2, D3.	TransferredEvent • primaryOldCall • secondaryOldCall • transferringDevice • transferredDevice • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1C2 D1/NR D3 *D2C3* *D2* *D2C1* *D3C3* *D3* *D3C2* *Null* *( )* *Transfer*	TransferredEvent • primaryOldCall • *secondaryOldCall* • transferringDevice • transferredDevice • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 *( )* D1 D3 *D2C3* *D2* *D2C1* *D3C3* *D3* *D3C2* *Connected* *( )* *Transfer*	TransferredEvent • primaryOldCall • *secondaryOldCall* • transferringDevice • transferredDevice • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D3C2 *( )* D1 D3/NR *D2C3* *D2* *D2C1* *D3C3* *D3* *D2C2* *Alerting* *( )* *Transfer*	*The result is that connections D1C1,D1C2, D2C1 and D3C2 are cleared. SecondaryOldCall parameter is only included if events have been reported for the connection identifier.*

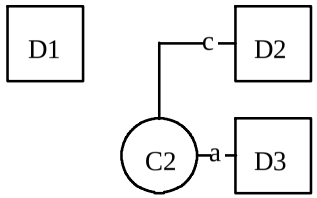
8.3 Single Step Transfer

Device D1 has an active call to device D2 and invokes the Single Step Transfer service to transfer the call to device D3. Device D1 does not participate in the new call and clears.

BEFORE



AFTER

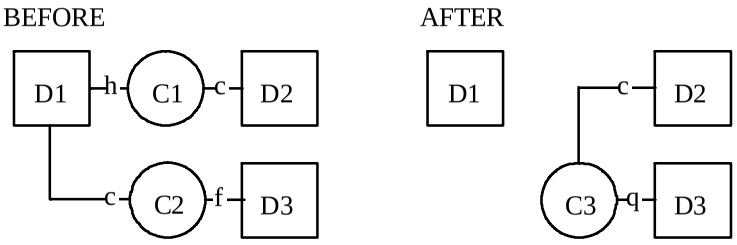


Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3	Comments
Single Step Transfer service is invoked on behalf of device D1.	SingleStepTransferRequest • activeCall • deviceToTransferTo • *transferToDeviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D1C1 D3 *( )* *( )* *( )* *( )* *( )*				
Single Step Transfer is accepted.	SingleStepTransferResult • *transferredCall* • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *extensions*	 *D3C2* *D2C2* *D2* *D2C1* *D3C2* *D3* *( )*				
The call between D1,D2 is replaced with an alerting call between D2,D3.	TransferredEvent • primaryOldCall • *secondaryOldCall* • transferringDevice • transferredDevice • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 *( )* D1/NR D3 *D2C2* *D2* *D2C1* *D3C2* *D3* *Null* *( )* *SingleStep Transfer*	TransferredEvent • primaryOldCall • *secondaryOldCall* • transferringDevice • transferredDevice • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 *( )* D1 D3 *D2C2* *D2* *D2C1* *D3C2* *D3* *Connected* *( )* *SingleStep Transfer*		*As there is no connection to D3 at this time it will not see the transferred event.*

Device D3 begins to ring.		DeliveredEvent • connection D3C2 • alertingDevice D3 • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*	DeliveredEvent • connection D3C2 • alertingDevice D3/NR • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Alerting* • *correlatorData* *( )* • *cause* *singleStep Transfer*	*This event reflects the state change for connection D3C2.*

8.4 Transfer on Busy

Device D1 has an active call to device D2 and invokes a Consultation call with device D3. Device D3 is busy but device D1 still transfers the call so that D2 camps-on to device D3. The scenario below illustrates the events once the transfer has been invoked.



Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3		Comments
Transfer Call service is invoked on behalf of device D1.	TransferCallRequest • heldCall • activeCall • *extensions*	 D1C1 D1C2 *( )*					
Acknowledgement.	TransferCallResult • *transferredCall* • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *extensions*	 *D3C3* *D2C3* *D2* *D2C1* *D3C3* *D3* *D3C2* *( )*					
The calls between D1,D2 and D1,D3 are released. The connections between D2, D1 and D3, D1 are replaced with a single queued call between D2, D3.	TransferredEvent • primaryOldCall • secondaryOldCall • transferringDevice • transferredDevice • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1C2 D1/NR D3 *D2C3* *D2* *D2C1* *D3C3* *D3* *D3C2* *Null* *( )* *Transfer*	TransferredEvent • primaryOldCall • *secondaryOldCall* • transferringDevice • transferredDevice • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 *( )* D1 D3 *D2C3* *D2* *D2C1* *D3C3* *D3* *D3C2* *Connected* *( )* *Transfer*	TransferredEvent • primaryOldCall • *secondaryOldCall* • transferringDevice • transferredDevice • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D3C2 *( )* D1 D3/NR *D2C3* *D2* *D2C1* *D3C3* *D3* *D3C2* *Failed* *( )* *Transfer*	

The call queues at device D3.		QueuedEvent • queuedConnection D3C3 • queue D3 • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NR • *numberedQueued* *( )* • *callsInFront* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *campON*	QueuedEvent • queuedConnection D3C3 • queue D3/NR • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NR • *numberedQueued* *( )* • *callsInFront* *( )* • *localConnectionInfo* *Queued* • *correlatorData* *( )* • *cause* *campON*	
Device D3 becomes free and begins to ring.		DeliveredEvent • connection D3C3 • alertingDevice D3 • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*	DeliveredEvent • connection D3C3 • alertingDevice D3/NR • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Alerting* • *correlatorData* *( )* • *cause* *newCall*	
Device D3 answers the call.		EstablishedEvent • establishedConnection D3C3 • answeringDevice D3 • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*	EstablishedEvent • establishedConnection D3C3 • answeringDevice D3/NR • callingDevice D2 • calledDevice D3 • lastRedirectionDevice NR • *originatingConnection* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*	

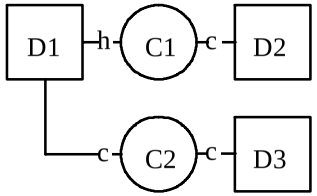
9 Conference Calls

Examples of simple Conference calls and more complex multi party Conference calls where individual parties join conferences, including a Single Step Conference.

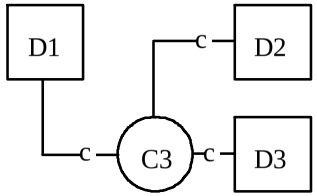
9.1 Conference Call Service

Device D1 has two calls, one active call and the other call on hold. This service provides the ability to conference the three devices into a single call.

BEFORE



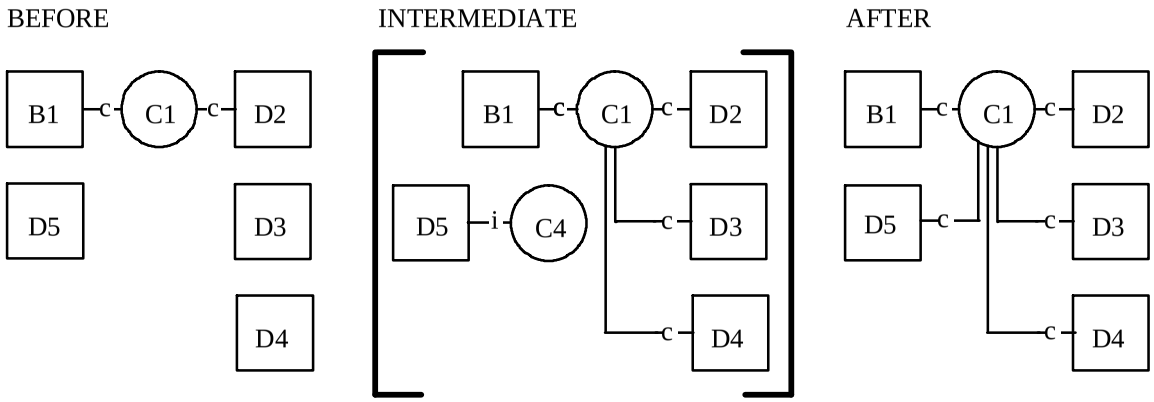
AFTER



Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3		Comments
Conference Call service is invoked on behalf of device D1.	ConferenceCallRequest • heldCall • activeCall • *extensions*	 D1C1 D1C2 *( )*					
Acknowledgement.	ConferenceCallResult • conferenceCall • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *extensions*	 D1C3 *D2C3* *D2* *D2C1* *D3C3* *D3* *D3C2* *( )*					
Conference established.	ConferencedEvent • primaryOldCall • secondaryOldCall • confController • addedParty • *newConnection* • *deviceID* • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1C2 D1/NR D3 *D1C3* *D1* *D2C3* *D2* *D2C1* *D3C3* *D3* *D3C2* *Connected* *( )* *newCall*	ConferencedEvent • primaryOldCall • *secondaryOldCall* • confController • addedParty • *newConnection* • *deviceID* • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 *( )* D1 D3 *D1C3* *D1* *D2C3* *D2* *D2C1* *D3C3* *D3* *D3C2* *Connected* *( )* *newCall*	ConferencedEvent • primaryOldCall • *secondaryOldCall* • confController • addedParty • *newConnection* • *deviceID* • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D3C2 *( )* D1 D3/NR *D1C3* *D1* *D2C3* *D2* *D2C1* *D3C3* *D3* *D3C2* *Connected* *( )* *newCall*	*The added party is the device representing the person who has just joined the call from the perspective of the participants.*

9.2 Multiple Party (Meet Me) Conference

Device B1 is a conference bridge. Each device that joins the conference dials the conference access number. All the devices in the conference bridge have the same call ID. This scenario represents the case when device D2 is the first to join the conference bridge B1 followed by another device, D5, joining into the conference. For clarity events for devices D3 & D4 are not shown below.



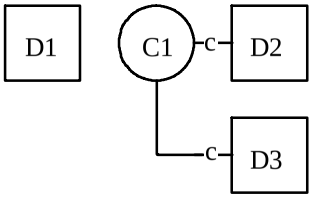
Activity	MONITORED DEVICE B1	MONITORED DEVICE D2	MONITORED DEVICE D5	Comments
Device B1 "answers" the call from device D2.	EstablishedEvent • establishedConnection D2C1 • answeringDevice B1 • callingDevice D2 • calledDevice B1 • lastRedirectionDevice NR • *originatingConnection ( )* • *localConnectionInfo Connected* • *correlatorData ( )* • *cause newCall*	EstablishedEvent • establishedConnection D2C1 • answeringDevice B1 • callingDevice D2 • calledDevice B1 • lastRedirectionDevice NR • *originatingConnection ( )* • *localConnectionInfo Connected* • *correlatorData ( )* • *cause newCall*		*For a simple "two" party call an established event is generated.*
	....previous call establishments and conferencing for devices D3 and D4 take place here....			
Device D5 dials into the conference bridge.			OriginatedEvent • originatedConnection D5C4 • callingDevice D5/NR • calledDevice B1 • *originatingDevice ( )* • *localConnectionInfo Connected* • *correlatorData ( )* • *cause newCall*	*Call IDs C2 and C3 were associated with the originated events for D3 and D4 originating calls to the conference bridge.*

Device B1 "answers" the call.	ConferencedEvent		ConferencedEvent		ConferencedEvent		*D5 has a cause because there is a new call ID.*
	• primaryOldCall	B1C1	• primaryOldCall	D2C1	• primaryOldCall	D5C4	
	• *secondaryOldCall*	*( )*	• *secondaryOldCall*	*( )*	• *secondaryOldCall*	*( )*	
	• confController	B1/NR	• confController	B1	• confController	B1	
	• addedParty	D5	• addedParty	D5	• addedParty	D5/NR	
	• *newConnection*	*B1C1*	• *newConnection*	*B1C1*	• *newConnection*	*B1C1*	
	• *deviceID*	*B1*	• *deviceID*	*B1*	• *deviceID*	*B1*	
	• *oldConnection*	*B1C1*	• *oldConnection*	*B1C1*	• *oldConnection*	*B1C1*	
	• *newConnection*	*D2C1*	• *newConnection*	*D2C1*	• *newConnection*	*D2C1*	
	• *deviceID*	*D2*	• *deviceID*	*D2*	• *deviceID*	*D2*	
	• *oldConnection*	*D2C1*	• *oldConnection*	*D2C1*	• *oldConnection*	*D2C1*	
	• *newConnection*	*D3C1*	• *newConnection*	*D3C1*	• *newConnection*	*D3C1*	
	• *deviceID*	*D3*	• *deviceID*	*D3*	• *deviceID*	*D3*	
	• *oldConnection*	*D3C1*	• *oldConnection*	*D3C1*	• *oldConnection*	*D3C1*	
	• *newConnection*	*D4C1*	• *newConnection*	*D4C1*	• *newConnection*	*D4C1*	
	• *deviceID*	*D4*	• *deviceID*	*D4*	• *deviceID*	*D4*	
	• *oldConnection*	*D4C1*	• *oldConnection*	*D4C1*	• *oldConnection*	*D4C1*	
	• *newConnection*	*D5C1*	• *newConnection*	*D5C1*	• *newConnection*	*D5C1*	
	• *deviceID*	*D5*	• *deviceID*	*D5*	• *deviceID*	*D5*	
	• *oldConnection*	*D5C4*	• *oldConnection*	*D5C4*	• *oldConnection*	*D5C4*	
	• *localConnectionInfo*	*Connected*	• *localConnectionInfo*	*Connected*	• *localConnectionInfo*	*Connected*	
	• *correlatorData*	*( )*	• *correlatorData*	*( )*	• *correlatorData*	*( )*	
	• *cause*	*( )*	• *cause*	*( )*	• *cause*	*newCall*	

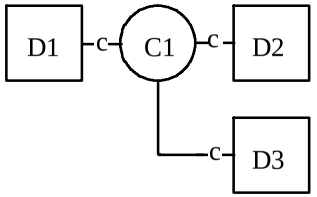
9.3 Single Step Conference

Device D2 and device D3 are involved in a call C1. The Single Step Conference service is invoked on behalf of device D1 which wishes to silently join the call. The three devices D1, D2 and D3 are then involved in a single call C1. The scenario begins at the point where device D1 begins to join the call. This scenario does not show device D1 being prompted to lift the handset or invoking the service from a feature button.

BEFORE



AFTER

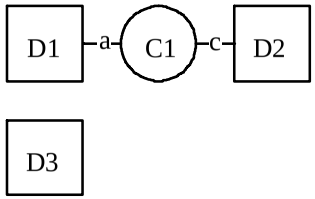


Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3		Comments
SingleStepConference service invoked on behalf of D1.	SingleStepConfRequest • activeCall • deviceToJoin • *participationType* • *joiningDeviceProfile* • *accountCode* • *authCode* • *correlatorData* • *extensions*	 D2C1 D1 *Silent* *( )* *( )* *( )* *( )* *( )*					*It is possible that some devices will be prohibited from joining calls.*
Acknowledgement.	SingleStepConfResult • conferencedCall • *extensions*	 D1C1 *( )*					
Conference now established.	ConferencedEvent • primaryOldCall • *secondaryOldCall* • confController • addedParty • *newConnection* • *deviceID* • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 *( )* D1/NR D1/NR *D1C1* *D1* *D2C1* *D2* *D2C1* *D3C1* *D3* *D3C1* *Connected* *( )* *silent Monitor*	ConferencedEvent • primaryOldCall • *secondaryOldCall* • confController • addedParty • *newConnection* • *deviceID* • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 *( )* D1 D1 *D1C1* *D1* *D2C1* *D2* *D2C1* *D3C1* *D3* *D3C1* *Connected* *( )* *silent Monitor*	ConferencedEvent • primaryOldCall • *secondaryOldCall* • confController • addedParty • *newConnection* • *deviceID* • *newConnection* • *deviceID* • *oldConnection* • *newConnection* • *deviceID* • *oldConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D3C1 *( )* D1 D1 *D1C1* *D1* *D2C1* *D2* *D2C1* *D3C1* *D3* *D3C1* *Connected* *( )* *silent Monitor*	*During a Silent SingleStepConference it is an implementation issue regarding the devices which see the ConferencedEvent.*

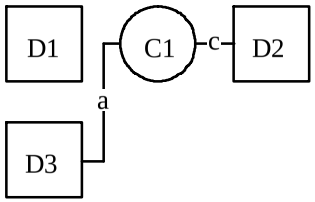
10 Divert Deflection Service

A call is in the process of being established between device D2 and device D1. When device D1 detects the incoming call it invokes the divert deflection service redirecting the incoming call to a new destination, device D3.

BEFORE



AFTER

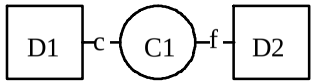


Activity	MONITORED DEVICE D1	MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
Divert deflection service invoked on behalf of device D1.	DivertCallRequest • callToBeDiverted D1C1 • newDestination D3 • *deviceProfile* *( )* • *correlatorData* *( )* • *extensions* *( )*			*The state of D1C1 may be connected, queued or alerting.*
Acknowledgement.	DivertCallResult • *extensions* *( )*			
Call to D1 gets diverted.	DivertedEvent • connection D1C1 • divertingDevice D1/NR • newDestination D3 • *localConnectionInfo* *Null* • *correlatorData* *( )* • *cause* *Redirected*			
Diverted call then alerts device D3.		DeliveredEvent • connection D3C1 • alertingDevice D3 • callingDevice D2/NR • calledDevice D1 • lastRedirectionDevice D1 • *originatingConnection* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *Redirected*	DeliveredEvent • connection D3C1 • alertingDevice D3/NR • callingDevice D2 • calledDevice D1 • lastRedirectionDevice D1 • *originatingConnection* *( )* • *localConnectionInfo* *Alerting* • *correlatorData* *( )* • *cause* *Redirected*	

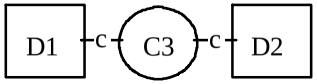
11 Call Completion Service

A call attempt between device D1 and device D2 results in a busy connection. The calling party, device D1, invokes the callback Call Completion service. The initial part of the scenario is as a Make Call which meets busy.

BEFORE



AFTER



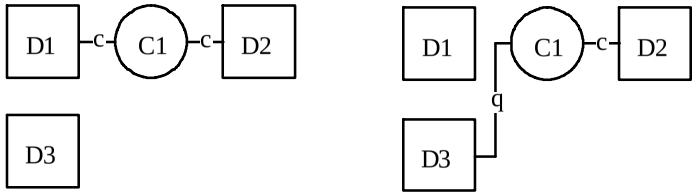
Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3	Comments
Device D2 is busy.	FailedEvent • failedConnection • failingDevice • calledDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2 D2 *Connected* *( )* *Busy*	FailedEvent • failedConnection • failingDevice • calledDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2/NR D2 *Failed* *( )* *Busy*		
The Call Completion service is invoked on behalf of device D1, the calling party.	CallCompletionRequest • callback • *extensions*	 D1C1 *( )*				
Acknowledgement.	CallCompletionResult • *extensions*	 *( )*				
Device D1 replaces the handset.	ConnectionClearedEvent • droppedConnection • releasingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR *Null* *( )* *callBack*	ConnectionClearedEvent • droppedConnection • releasingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1 *Failed* *( )* *callBack*		
Failed connection D2C1 also clears.			ConnectionClearedEvent • droppedConnection • releasingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2/NR *Null* *( )* *normal Clearing*		
Device D2 sometime later clears from their active call C2.			ConnectionClearedEvent • droppedConnection • releasingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D2C2 D2/NR *Null* *( )* *normal Clearing*		
Device D1 prompted to lift handset for callback.	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C3 *Initiated* *callBack*				

Device D1 lifts handset - callback now proceeds.	OriginatedEvent • originatedConnection D1C3 • callingDevice D1/NR • calledDevice D2 • *originatingDevice ( )* • *localConnectionInfo Connected* • *correlatorData ( )* • *cause callBack*			
Device D2 begins to ring.	DeliveredEvent • connection D2C3 • alertingDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NR • *originatingConnection ( )* • *localConnectionInfo Connected* • *correlatorData ( )* • *cause callBack*	DeliveredEvent • connection D2C3 • alertingDevice D2/NR • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NR • *originatingConnection ( )* • *localConnectionInfo Alerting* • *correlatorData ( )* • *cause callBack*		
D2 connects to the call.	EstablishedEvent • establishedConnection D2C3 • answeringDevice D2 • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NR • *originatingConnection ( )* • *localConnectionInfo Connected* • *correlatorData ( )* • *cause callBack*	EstablishedEvent • establishedConnection D2C3 • answeringDevice D2/NR • callingDevice D1 • calledDevice D2 • lastRedirectionDevice NR • *originatingConnection ( )* • *localConnectionInfo Connected* • *correlatorData ( )* • *cause callBack*		

12 A Manually Invoked Call Park

In this scenario, D1 and D2 are in an established call. Device D1 decides to put the call in park (illustrated by device D3). This operation can be done both manually or by using a park button on a feature phone.

BEFORE AFTER



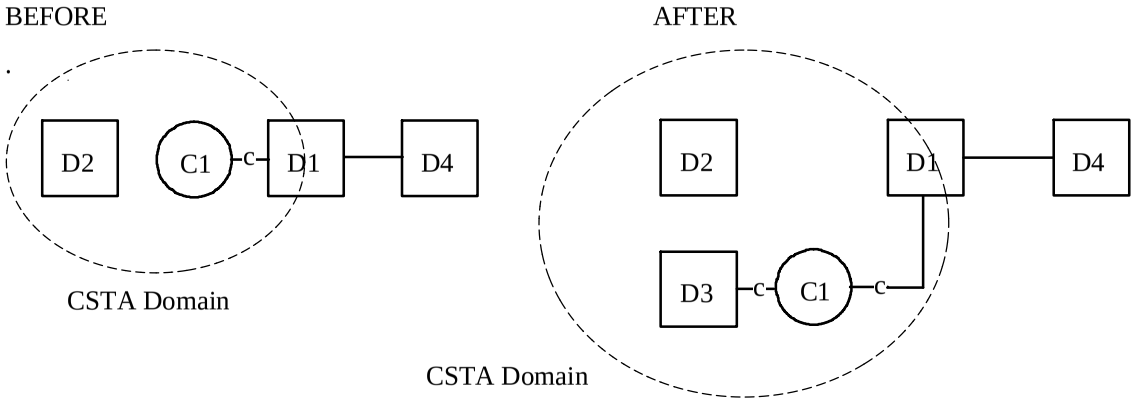
Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3		Comments
A Call Park is initiated by device D1. Call to device D2 is placed on hold.	HeldEvent • heldConnection • holdingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR *Hold* *( )* *( )*	HeldEvent • heldConnection • holdingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1 *Connected* *( )* *( )*			*If the phone has a Call Park feature button then this event can be omitted.*
Indication that device D1 has invoked a service .	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C2 *Initiated* *newCall*					
Call is queued at device D3.	QueuedEvent • queuedConnection • queue • callingDevice • calledDevice • lastRedirectionDevice • *numberedQueued* • *callsInFront* • *localConnectionInfo* • *correlatorData* • *cause*	 D3C1 D3 D2 D3 NR *( )* *( )* *Connected* *( )* *Park*	QueuedEvent • queuedConnection • queue • callingDevice • calledDevice • lastRedirectionDevice • *numberedQueued* • *callsInFront* • *localConnectionInfo* • *correlatorData* • *cause*	 D3C1 D3 D2 D3 NR *( )* *( )* *Connected* *( )* *Park*	QueuedEvent • queuedConnection • queue • callingDevice • calledDevice • lastRedirectionDevice • *numberedQueued* • *callsInFront* • *localConnectionInfo* • *correlatorData* • *cause*	 D3C1 D3/NR D2 D3 NR *( )* *( )* *Queued* *( )* *Park*	
Device D2 is connected to the Park device.	ConnectionClearedEvent • droppedConnection • releasingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR *Null* *( )* *Park*	ConnectionClearedEvent • droppedConnection • releasingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1 *Connected* *( )* *Park*			
Device D1 replaces handset and the feature invocation call C2, clears.	ConnectionClearedEvent • droppedConnection • releasingDevice • *localConnectionInfo* • *correlatorData* • *cause*	 *D1C2* *D1* *Null* *( )* *normal Clearing*					

13 Route Services

This section of scenarios includes examples where the switch asks the application how to route a call. Routeing at a particular device is enabled by using the Set Feature service as shown below. Only the routeing dialogue is shown - not all the associated call events.

13.1 Route Request Service

Switch invokes the Route Request service for an emergency call of high priority to which a reply is received and accepted by the switch. All the following entries are service requests.

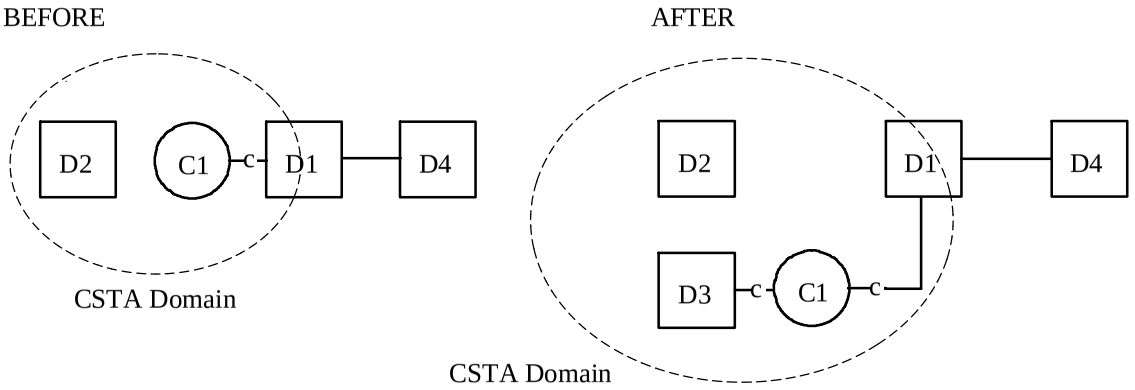


Activity	ROUTEING DEVICE D1		ROUTEING DEVICE D2	ROUTEING DEVICE D3	Comments
The Set Feature service is invoked to enable routeing on behalf of device D1.	SetFeatureRequest • device • feature • *deviceProfile* • *extensions*	 D1 enable Routing *( )* *( )*			*In this example device D1 represents a trunk.*
Acknowledgement.	SetFeatureResult • *extensions*	 *( )*			
Switch initiates a RouteRequest when a call involves device D1.	RouteRequest • crossRefIdentifier • currentRoute • *callingDevice* • *routingDevice* • *routedCall* • *routeSelAlgorithm* • *priority* • *deviceProfile* • *correlatorData* • *extensions*	 Ref1 D2 *D4* D1/NR *D1C1* *emergency* *TRUE* *( )* *( )* *( )*			

Computer responds with an alternative route.	RouteSelectRequest • crossRefIdentifier Ref1 • routeSelected D3 • *remainRetry* *( )* • *deviceProfile* *( )* • *routeUsedReq* *( )* • *correlatorData* *( )* • *extensions* *( )*			
Switch ends re-route session.	RouteEndRequest • crossRefIdentifier Ref1 • *errorValue* *( )*			

13.2 Route Used Service

Switch invokes the Route Request service for an emergency call of high priority to which a reply is received and accepted by the switch. In this case the computer requests that it is informed of the final route used by the switch. All the following entries are Service Requests. Routeing is already enabled.



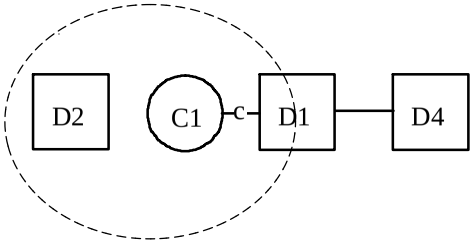
Activity	ROUTEING DEVICE D1		ROUTEING DEVICE D2	ROUTEING DEVICE D3	Comments
Switch invokes the RouteRequest service when a call involves device D1.	RouteRequest • crossRefIdentifier • currentRoute • *callingDevice* • *routingDevice* • *routedCall* • *routeSelAlgorithm* • *priority* • *deviceProfile* • *correlatorData* • *extensions*	 Ref1 D2 *D4* *D1/NR* *D1C1* *emergency* *TRUE* *( )* *( )* *( )*			*In this example device D1 represents a trunk.*
Computer responds with an alternative route.	RouteSelectRequest • crossRefIdentifier • routeSelected • *remainRetry* • *deviceProfile* • *routeUsedReq* • *correlatorData* • *extensions*	 Ref1 D3 *( )* *( )* *TRUE* *( )* *( )*			
Switch notifies computer of the route used.	RouteUsedRequest • crossRefIdentifier • routeUsed • *callingDevice* • *domain* • *correlatorData* • *extensions*	 Ref1 D3 *( )* *( )* *( )* *( )*			*The route used could be different if the switch decides to ignore the route provided by the computer.*

Switch ends re-route session.	RouteEndRequest • crossRefIdentifier Ref1 • *errorValue* *( )* • *extensions* *( )*			

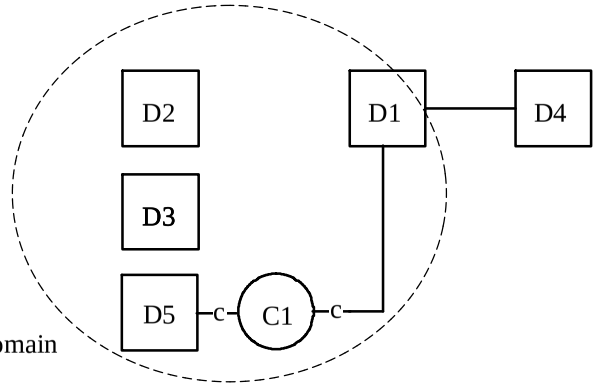
13.3 Re-Route Service

Switch invokes the Route Request service for an emergency call of high priority to which a reply is received but the switch requests an alternative. The computer requests that it is informed of the final route used by the switch. All the entries below are Service Requests. Routeing is already enabled.

BEFORE



AFTER



Activity	ROUTEING DEVICE D1		ROUTEING DEVICE D2	ROUTEING DEVICE D3	Comments
Switch initiates RouteRequest.	RouteRequest • crossRefIdentifier • currentRoute • *callingDevice* • *routingDevice* • *routedCall* • *routeSelAlgorithm* • *priority* • *deviceProfile* • *correlatorData* • *extensions*	 Ref1 D2 *D4* *D1/NR* *D1C1* *emergency* *TRUE* *( )* *( )* *( )*			
Computer responds with an alternative route.	RouteSelectRequest • crossRefIdentifier • routeSelected • *remainRetry* • *deviceProfile* • *routeUsedReq* • *correlatorData* • *extensions*	 Ref1 D3 *( )* *( )* *TRUE* *( )* *( )*			
Switch asks for an alternative.	Re-routeRequest • crossRefIdentifier • *extensions*	 Ref1 *( )*			

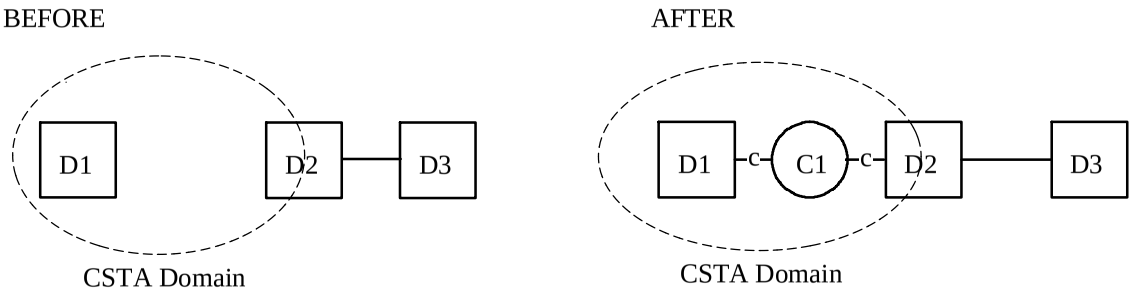
Computer responds with another alternative route.	RouteSelectRequest • crossRefIdentifier Ref1 • routeSelected D5 • *remainRetry* *( )* • *deviceProfile* *( )* • *routeUsedReq* *TRUE* • *correlatorData* *( )* • *extensions* *( )*			
Switch notifies computer of the route used.	RouteUsedRequest • crossRefIdentifier Ref1 • routeUsed D5 • *callingDevice* *( )* • *domain* *( )* • *correlatorData* *( )* • *extensions* *( )*			*The route used could be different if the switch decides to ignore the route provided by the computer.*
Switch ends re-route session.	RouteEndRequest • crossRefIdentifier Ref1 • *errorValue* *( )* • *extensions* *( )*			

14 Incoming Calls

This section includes an example of an incoming call from outside of the CSTA domain handled by a trunk within the CSTA domain and an example of an incoming call to an ACD queue.

14.1 Successful incoming Call

In this scenario an incoming trunk, device D2 (the originating device), is initiating a call on behalf of device D3 (the calling device) to a destination within the CSTA sub-domain, device D1 (the called device).



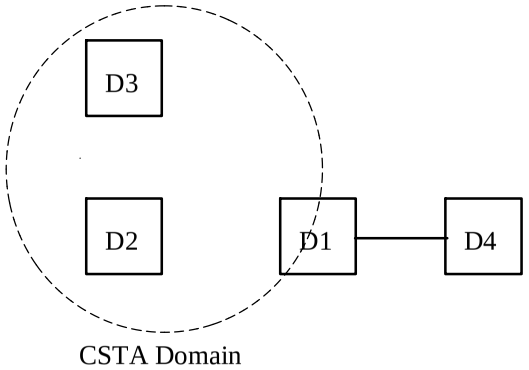
Activity	MONITORED DEVICE D1	MONITORED DEVICE D2	MONITORED DEVICE D3	Comments
Device D2 goes "off-hook".		ServiceInitiatedEvent • initiatedConnection D2C1 • *localConnectionInfo* *Initiated* • *cause* *newCall*		*This event can be generated for all types of trunks excluding en-bloc dialling (i.e by an ISDN trunk).*
Device D2 completes dialling.		OriginatedEvent • originatedConnection D2C1 • callingDevice D3/NK • calledDevice D1 • *originatingDevice* *D2* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*		*This event can be generated for ISDN en-bloc dialling. Information about D3 may exist in some types of trunks like ISDN or T1 with support of ANI.*
Device D1 begins to ring and D2/D3 listens to ringing tone.	DeliveredEvent • connection D1C1 • alertingDevice D1/NR • callingDevice D3/NK • calledDevice D1 • lastRedirectionDevice NR • *originatingConnection* *D2C1* • *localConnectionInfo* *Alerting* • *correlatorData* *( )* • *cause* *newCall*	DeliveredEvent • connection D1C1 • alertingDevice D1 • callingDevice D3/NK • calledDevice D1 • lastRedirectionDevice NR • *originatingConnection* *D2C1* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *newCall*		*Information about D3 may exist in some types of trunks like ISDN or T1 with support of ANI.*

Device D1 answers the call.	EstablishedEvent • establishedConnection • answeringDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1/NR D3/NK D1 NR *D2C1* *Connected* *( )* *newCall*	EstablishedEvent • establishedConnection • answeringDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D1 D3/NK D1 NR *D2C1* *Connected* *( )* *newCall*		*Information about D3 may exist in some types of trunks like ISDN or T1.*

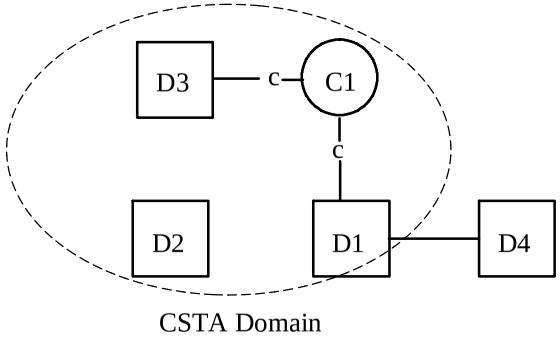
14.2 Incoming call to ACD with no available agents

Incoming call to an ACD distribution mechanism (device D2) where the call awaits distribution to a free agent. While queuing, the trunk (device D1) may be connected with a recorded announcement (events not shown). The recording may repeat itself or change to a different one, several times while the call is queuing at the ACD distribution mechanism. When an agent becomes free (device D3) the call is delivered to that device.

BEFORE



AFTER



Activity	MONITORED DEVICE D1		MONITORED DEVICE D2		MONITORED DEVICE D3	Comments
Device D1 goes "off-hook".	ServiceInitiatedEvent • initiatedConnection • *localConnectionInfo* • *cause*	 D1C1 *Initiated* *newCall*				
Device D1 completes dialling.	OriginatedEvent • originatedConnection • callingDevice • calledDevice • *originatingDevice* • *localConnectionInfo* • *correlatorData* • *cause*	 D1C1 D4/NK D2 *D1/NR* *Connected* *( )* *newCall*				
The call begins to alert at the queue.	DeliveredEvent • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2 D4 D2 NR *D1C1* *Connected* *( )* *entering Distribution*	DeliveredEvent • connection • alertingDevice • callingDevice • calledDevice • lastRedirectionDevice • *originatingConnection* • *localConnectionInfo* • *correlatorData* • *cause*	 D2C1 D2/NR D4 D2 NR *D1C1* *Alerting* *( )* *entering Distribution*		*Information about D4 may exist in some types of trunks like ISDN or T1 with support of ANI.*

Call is queued in Q1 represented by device D2.	QueuedEvent • queuedConnection D2C1 • queue Q1 • callingDevice D4 • calledDevice D2 • lastRedirectionDevice NR • *numberedQueued* *( )* • *callsInFront* *( )* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *noAvailable Agents*	QueuedEvent • queuedConnection D2C1 • queue Q1 • callingDevice D4 • calledDevice D2 • lastRedirectionDevice NR • *numberedQueued* *( )* • *callsInFront* *( )* • *localConnectionInfo* *Queued* • *correlatorData* *( )* • *cause* *noAvailable Agents*		*Q1 may identify an ACD group or the ACD queuing mechanism. A call may reside within multiple queues.*
The queued call gets diverted to device D3. D3 represents an agent which is now free.		DivertedEvent • connection D2C1 • divertingDevice D2/NR • newDestination D3 • *localConnectionInfo* *Null* • *correlatorData* *( )* • *cause* *Distributed*		*An annoucement can be provided prior to this event for which events may be generated.*
Device D3 is alerted.	DeliveredEvent • connection D3C1 • alertingDevice D3 • callingDevice D4 • calledDevice D2 • lastRedirectionDevice D2 • *originatingConnection* D1C1 • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *Distributed*		DeliveredEvent • connection D3C1 • alertingDevice D3/NR • callingDevice D4 • calledDevice D2 • lastRedirectionDevice D2 • *originatingConnection* D1C1 • *localConnectionInfo* *Alerting* • *correlatorData* *( )* • *cause* *Distributed*	
Device D3 answers the call.	EstablishedEvent • establishedConnection D3C1 • answeringDevice D3 • callingDevice D4 • calledDevice D2 • lastRedirectionDevice D2 • *originatingConnection* *D1C1* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *Distributed*		EstablishedEvent • establishedConnection D3C1 • answeringDevice D3/NR • callingDevice D4 • calledDevice D2 • lastRedirectionDevice D2 • *originatingConnection* *D1C1* • *localConnectionInfo* *Connected* • *correlatorData* *( )* • *cause* *Distributed*	

This ECMA Technical Report TR/68 is available free of charge from:

ECMA
114 Rue du Rhône
CH-1204 Geneva
Switzerland

Fax: +41 22 849.60.01
Internet: helpdesk@ecma.ch

This Technical Report can also be downloaded as file T068-DOC.EXE or file T068-PSC.EXE from ECMANEWS