Genesys Events and Models Reference

System Level Guides Current

12/29/2021

Table of Contents

Genesys Events and Models Reference

Welcome to the *Genesys Events and Models Reference.* This document introduces you to many of the call and interaction events and models that you may encounter in a Genesys deployment. This document is valid for the 7.x, 8.x, and 9.x releases of products related to these models.

In brief, you will find the following information in this guide:

- A full list of call and other interaction events and their descriptions.
- A collection of common call and other interaction models and flows.
- Some specialized information on call and other interaction state.

About this Document

Intended Audience

This guide is intended for application developers who are familiar with Genesys deployments and need to do gain greater insight into their workings. Experienced system administrators might also use this *Reference* for advanced configuration issues. For instance, you might use this *Events and Models Reference* and its sample call models to help you handle and anticipate events for a custom application you are designing to work in a Genesys deployment. Or you might look at the important values associated with certain events in a model, and use that information to specially configure a routing strategy.

This document provides detailed information on the workings of interactions in a Genesys environment.

It assumes that you have a basic understanding of:

- The underlying concepts of CTI technology.
- A familiarity with the workings of Genesys Interactions.
- Genesys environment deployment and operation.
- Your own Genesys configurations.

You may also find that a familiarity with messaging-compliant programming gives you greater insight into issues as you read this document. A solid understanding of client-server implementations is also useful.

Usage Guidelines

The Genesys developer materials outlined in this document are intended to be used for the following purposes:

- Creation of agent desktop applications associated with Genesys software implementations.
- Server-side integration between Genesys software and third-party software.
- Creation of a specialized client application specific to customer needs.

The Genesys software functions available for development are clearly documented. No undocumented functionality is to be utilized without Genesys's express written consent.

The following Use Conditions apply in all cases for developers employing the Genesys developer materials outlined in this document:

1. Possession of interface documentation does not imply a right to use by a third party. Genesys conditions for use, as outlined below or in the *Genesys Developer Program Guide,* must be met.
2. This interface shall not be used unless the developer is a member in good standing of the Genesys Interacts program or has a valid Master Software License and Services Agreement with Genesys.
3. A developer shall not be entitled to use any licenses granted hereunder unless the developer's organization has met or obtained all prerequisite licensing and software as set out by Genesys.
4. A developer shall not be entitled to use any licenses granted hereunder if the developer's organization is delinquent in any payments or amounts owed to Genesys.
5. A developer shall not use the Genesys developer materials outlined in this document for any general application development purposes that are not associated with the above-mentioned intended purposes for the use of the Genesys developer materials outlined in this document.
6. A developer shall disclose the developer materials outlined in this document only to those employees who have a direct need to create, debug, and/or test one or more participant-specific objects and/or software files that access, communicate, or interoperate with the Genesys API.
7. The developed works and Genesys software running in conjunction with one another (hereinafter referred to together as the "integrated solutions") should not compromise data integrity. For example, if both the Genesys software and the integrated solutions can modify the same data, then modifications by either product must not circumvent the other product's data integrity rules. In addition, the integration should not cause duplicate copies of data to exist in both participant and Genesys databases, unless it can be assured that data modifications propagate all copies within the time required by typical users.
8. The integrated solutions shall not compromise data or application security, access, or visibility

restrictions that are enforced by either the Genesys software or the developed works.

9. The integrated solutions shall conform to design and implementation guidelines and restrictions described in the *Genesys Developer Program Guide* and Genesys software documentation. For example:
   - The integration must use only published interfaces to access Genesys data.
   - The integration shall not modify data in Genesys database tables directly using SQL.
   - The integration shall not introduce database triggers or stored procedures that operate on Genesys database tables.

Any schema extension to Genesys database tables must be carried out using Genesys Developer software through documented methods and features.

The Genesys developer materials outlined in this document are not intended to be used for the creation of any product with functionality comparable to any Genesys products, including products similar or substantially similar to Genesys's current general-availability, beta, and announced products.

Any attempt to use the Genesys developer materials outlined in this document or any Genesys Developer software contrary to this clause shall be deemed a material breach with immediate termination of this addendum, and Genesys shall be entitled to seek to protect its interests, including but not limited to, preliminary and permanent injunctive relief, as well as money damages.

Genesys Events and Models Overview

This *Genesys Events and Models Reference* provides you with a large collection of two different types of information: descriptions of Genesys events and illustrations of Genesys interaction models. Generally, you need to know about Genesys events and standard interaction models if you are developing custom applications for integration with Genesys systems, or if you are trying to gain greater insight into the flow of interactions in your environment.

Genesys Events topics describe everything from the names and descriptions of events, to the attributes that attend events, to the definitions of event substates. Based on the history of how this information has been presented in the past in various documents, event information may differ from topic to topic.

Genesys Interaction Models topics describe a selected list of call/interaction models. This information is also wide ranging. Based on the history of how this information has been presented in the past in various documents, model details may differ from topic to topic.

In both parts of this document, topics are organized according to the type of event or model being described. So, both parts have specific sections on voice-based issues that center on T-Library's generation of events and how calls are routed in a contact center.

Servers and Their Events and Models

When Genesys Servers start up and when they process interactions, they repeatedly perform certain tasks. This manual presents examples of these tasks in the form of events the servers generate and models that show how the components interact. You can study these events and models to:

- Get a better idea of how the various portions of your system work.
- Troubleshoot your system by inspecting the logs that record interactions of the type exemplified in this document.
- Understand what functions must be performed by a custom-built component that you want to design.

Events

Genesys servers generate events. These events can both be in response to requests that other components make of those servers, or they can be unsolicited (Genesys servers are programmed to notify clients for any number of reasons). Events that arrive in response to a request have an identifier you can use to link the two.

> **Important**
>
> Event names may differ slightly across Genesys SDK products and technologies. The names you see in this document correspond generally to the names you will see when using the SDKs. For the authoritative names of events used by your SDK, refer to the *API Reference* of the SDK your are using.

This document attempts to provide enough information about each event you may encounter so that you can intelligently handle events in your customized code, but also so you can better understand issues that arise in your environment. Reviewing log files is far easier when you have an understanding of the implications of given events.

Each event has a number of attributes associated with it to give you more information on the process that is occurring. A given event's attributes are not static. Depending on the request or environment, certain attributes may or may not be present. For completeness, this book documents all attributes that you might encounter with a given event.

Models

Call models and interaction flows serve a number of important purposes. First, no development for the core Genesys servers can take place unless there is a general understanding of how a given call scenario, for instance, should proceed in a contact center. The collections of interaction flows presented in this *Reference* represents the most common scenarios that Genesys software users encounter.

Elements in Call and Interaction Models

Events as depicted in the models are *protocol-specific*. That is, a given library issues its own events according to its own definitions (though some events from different libraries have similar names). The ordering and naming of events passed between components (messaging) in these models makes use of the various protocols being described. These include, but are not limited to:

- **T-Library (the `TLIB` protocol):** The core library for use in processing all voice-based interactions.
- **Multimedia Interaction Management (the `ITX` and `ESP` protocols):** The means by which Interaction Server processes its non-voice interactions.
- **Multimedia Reporting protocol:** This is Genesys Multimedia's own reporting protocol, different from

the one used by other Genesys applications that are reporting-specific.

- **Universal Routing Server's (URS) use of a subset of T-Library events:** Interaction Server uses these events to communicate with URS. When these T-Library events occur in the model diagrams, the equivalent Interaction Management protocol event is also shown. (The full collection of T-Library events, available in T-Library Events, is the authoritative list of T-Library events.)

Important

For events and models of protocols not listed here (for instance, STATLIB or CONFIGLIB), refer to the API Reference (Statistics or Configuration Platform SDK API Reference for .NET or Java, in these cases) of the SDK you are using.

Diagrams show time along the vertical axis, moving from top to bottom. Participants are arranged along the horizontal axis.

Interactions

The term *interaction* in a Genesys context has the following sense: an attempted communication between a customer and a contact center, in either direction. The attempt may be successful or not; it has a media type; it may give rise to other interactions (as when an incoming email gives rise to an automatic acknowledgement). It may belong to a series of related interactions, known as a *thread.* Interaction properties are generally recorded in a database.

Participants

Participants are software components that send and receive messages. The participants in the models in this book include the following:

- T-Server and IVR Server
- Switch and IVR
- Interaction Server
- Media server (E-mail Server Java, Chat Server, or a custom application)
- Agent application
- Universal Routing Server
- Reporting engine (Stat Server or ICON)

Protocol-Specific Issues

In general, this manual attempts to unify the concept of events and models across a number of disparate Genesys components. This allows you to move from component to component with one reference as your guide for how interactions are processed. However, you may find that you want more information about specific protocols. In each case, the authoritative collection of function calls and events for a given server's protocol is available from its corresponding Platform SDK API

Reference. (See the Platform SDK API Reference, .NET or Java for the server that interests you.)

Voice

The characteristic feature of events related to voice interactions is T-Server's use of the TEvent data structure to return the results of requests sent by applications using T-Library functions or as notification that there has been some activity on a monitored object.

Some examples of event member values include:

- `EventRegistered`—an application has registered a DN.
- `EventUnregistered`—an application has unregistered a DN.
- `EventAgentLogin`—an agent has logged in to an ACD group.
- `EventAgentLogout`—an agent has logged out of ACD.
- `EventAgentReady`—an agent is ready to receive ACD calls.

T-Library Events has detailed information on the TEvent data structure.

While this document presents a full representation of the TEvent structure, certain of its members are reserved for internal use only.

Genesys Events

Genesys Events section of this document contains specific information about Genesys events. This event information appears in the following sections:

- T-Library Events contains a list and description of each T-Library event that T-Servers generate, including the attributes that accompany each event. It also includes the following topics:
    - Agent States and Work Modes
    - Unified Call-Party States
    - Event Attributes
    - TEvent Structure
- T-Library Call-Based Notifications contains information on the TLIB events that relate to a call (as opposed to a DN or device).
- ISCC Transaction Monitoring describes how to work with ISCC regarding issues such as multi-site call transfer, inter-site call linkage, call overflow, and other, possibly non-call–related, multi-site operations.
- T-Library Unstructured Data provides information on how TLIB handles UserData, Extensions, and Reasons in events in a Genesys system.
- IVR Protocol Messages offers detailed information about IVR Server events.

T-Library Events

This section lists the T-Library Events that developers can expect to see while working with a Genesys implementation. Each event listed here is identified with a description, the contents of the event (presented in table format as a list of the attributes associated with it), and an example of where the event is likely to be encountered during a call flow. The end of this section includes general information about various event-related issues, including an Agent-State Diagram, definitions of event attributes (including, for the reference ID attribute, a table of the relationships between requests and events), and a description of the TEvent structure.

Event contents are presented as the collection of attributes associated with each event, as well as an indication of that attribute's *Type*. For the purposes of this section, Type has one of two values: *Mandatory* or *Optional*. Here Type refers to the presence of the attribute at the time of the generation of its event, and not to a characteristic of the attribute itself.

Attribute Type

- Mandatory—Indicates that this autoboot is always present when its associated event occurs.
- Optional—Indicates that this attribute may or may not be present when the associated event occurs.

List of T-Library Events

- **General Events**
    - EventServerConnected
    - EventServerDisconnected
    - EventLinkConnected
    - EventLinkDisconnected
- **Registration Events**
    - EventRegistered
    - EventUnregistered
- **Call-Handling and Transfer/Conference Events**
    - EventDialing
    - EventRinging
    - EventEstablished
    - EventAbandoned
    - EventDestinationBusy
    - EventDiverted
    - EventHeld

    - EventCallInfoChanged
    - EventPrivateInfo
    - EventUserEvent
    - EventPrimaryChanged
    - EventRestoreConnection
    - EventHardwareError
    - EventResourceAllocated (Obsolete—No Longer Supported)
    - EventResourceFreed (Obsolete—No Longer Supported)
- **Negative-Response Events**
    - EventError

Agent States, Call-Party States, Work Modes

The T-Library Events section also includes the following topics:

- Agent States and Work Modes
- Unified Call-Party States
- Event Attributes
- TEvent Structure

General Events

EventServerConnected

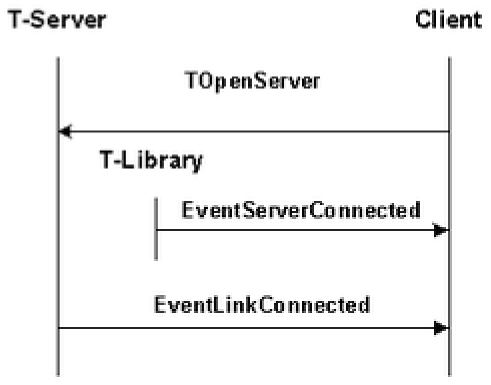
Description

The communication session with the server in question has been opened. This event is generated on the client side and only in asynchronous communication mode.

Contents

Event Attribute	Type
Event	Mandatory
Server	Mandatory
Extensions	Optional

Example



EventServerConnected Feature Example

EventServerDisconnected

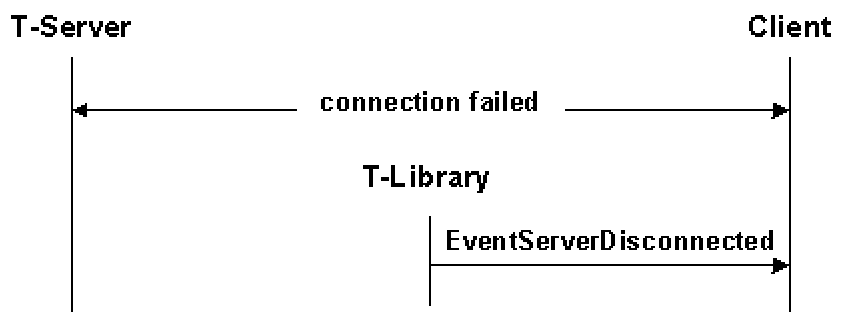
Description

The communication session with the server in question has failed. T-Library generates this event as soon as it recognizes that the connection with T-Server has been lost.

Contents

Event Attribute	Type
Event	Mandatory
Server	Mandatory
Extensions	Optional

Example



EventServerDisconnected Feature Example

EventLinkConnected

Description

This event is generated in two ways: either as a notification that the connection between the switch and T-Server, through the CTI link, has been restored, or as a response to the `TOpenServerEx()` and `TOpenServer()` functions (that is, after connecting to T-Server) to indicate the current CTI-link status.

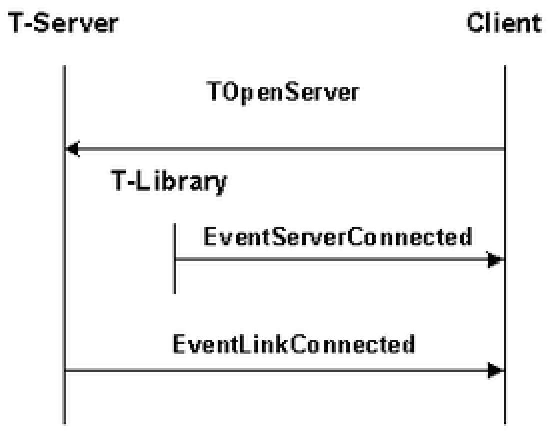
If you are working with a pair of T-Servers with Hot Standby mode in a high-availability environment, this event is also generated during T-Server switchover.

Contents

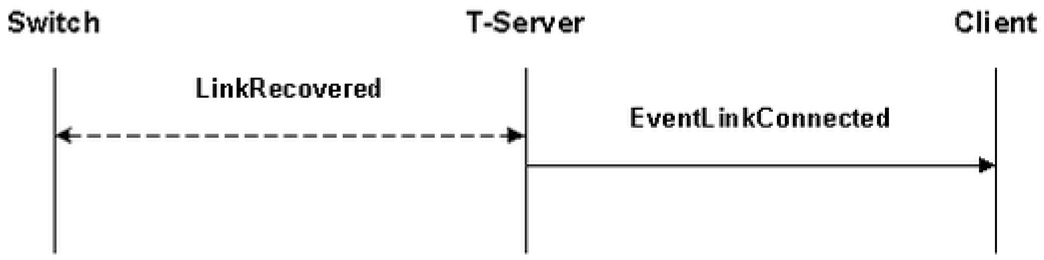
Event Attribute	Type
ApplicationName	Optional
Event	Mandatory
Server	Mandatory
SessionID	Optional

Event Attribute	Type
CustomerID	Optional
time	Mandatory
Extensions	Optional

Examples



EventLinkConnected Feature, Example 1



EventLinkConnected Feature, Example 2

EventLinkDisconnected

Description

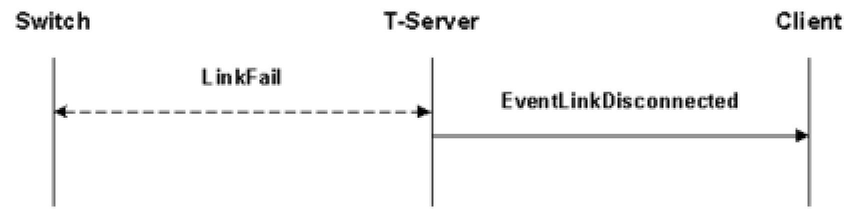
This event is generated as a notification that the connection between the switch and T-Server, through the CTI link, has been severed.

If you are working with a pair of T-Servers with Hot Standby mode in a high-availability environment, this event is also generated during T-Server switchover.

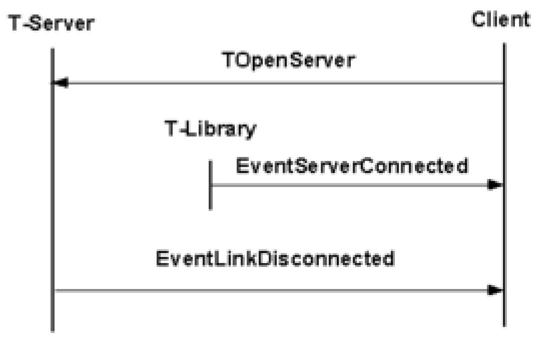
Contents

Event Attribute	Type
ApplicationName	Optional
SessionID	Optional
CustomerID	Optional
ErrorCode	Optional
ErrorMessage	Optional
Event	Mandatory
Server	Mandatory
time	Mandatory
Extensions	Optional

Examples



EventLinkDisconnected Feature, Example 1



EventLinkDisconnected Feature, Example 2

Registration Events

EventRegistered

Description

The application has been registered to send requests and receive events regarding the telephony object specified by `ThisDN`.

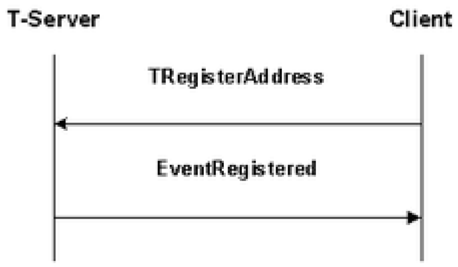
Contents

Event Attribute	Type
AgentID [a]	Optional
CustomerID	Optional
Event	Mandatory
Extensions [b]	Mandatory
InfoStatus.AddressType	Optional
InfoType=AddressInfoAddressType	Mandatory
ReferenceID	Mandatory
Server	Mandatory
ThisDN	Mandatory
time	Mandatory

a. The `AgentID` attribute can only be present for objects of `AddressTypePosition` or `AddressTypeDN`. See the contents of Extensions.
b. The `Extensions` attribute contains the same information as in EventAddressInfo:

- For objects of `AddressTypePosition` or `AddressTypeDN`, see the corresponding `InfoType=TAddressInfoDNStatus`.
- For objects of `AddressTypeQueue`, `AddressTypeRouteDN`, or `AddressTypeRouteQueue`, see `InfoType=TAddressInfoQueueStatus`.

Example



EventRegistered Feature Example

EventUnregistered

Description

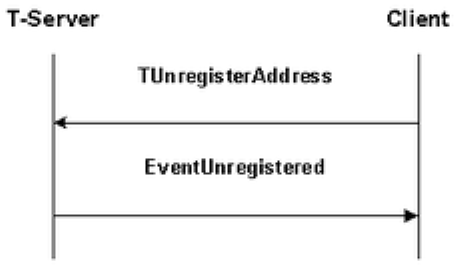
The application's registration to send requests and receive events regarding the telephony object specified by `ThisDN` has been canceled.

Contents

Event Attribute	Type
CustomerID	Optional
ErrorCode	Optional
ErrorMessage	Optional
Event	Mandatory
ReferenceID [a]	Optional
Server	Mandatory
ThisDN	Mandatory
time	Mandatory
Extensions	Optional

a. The `ReferenceID` attribute is mandatory in most cases, but not present if the switch configuration has been changed while the CTI link was down or if T-Server configuration has been changed dynamically through Configuration Manager. Meanwhile, the cause of unregistration is specified by the attributes `ErrorCode` and `ErrorMessage`.

Example



EventUnregistered Feature Example

Call-Handling and Transfer/Conference Events

EventDialing

Description

An attempt to make a call on behalf of the telephony object specified by `ThisDN` is in progress.

Contents

Event Attribute	Type
AgentID	Optional
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallType [a]	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN [b]	Optional
OtherDNRole [b]	Optional
OtherQueue	Optional
OtherTrunk	Optional
PreviousConnID [c]	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
ThisDNRole	Mandatory

Event Attribute	Type
ThisQueue [d]	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. `CallType` may be `Unknown`.
b. `OtherDN` may be either a dialed number or not present if T-Server has no information about the other party. `OtherDNRole` appears if the attribute `OtherDN` is present.
c. `PreviousConnID` must appear if the value of `CallType` is `Consult`.
d. `ThisQueue` must appear in predictive dialing and be equal to `ThisDN`.

Examples

See the example after EventRinging.

EventRinging

Description

A call has been delivered to the telephony object specified by `ThisDN`.

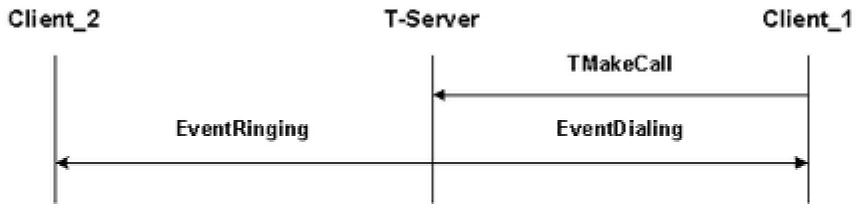
Contents

Event Attribute	Type
AgentID	Optional
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallState	Mandatory
CallType	Mandatory
Cause	Optional
ConnID	Mandatory
CollectedDigits	Optional
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional

Event Attribute	Type
NetworkNodeID	Optional
OtherDN	Optional
OtherDNRole	Optional
OtherQueue	Optional
OtherTrunk	Optional
PreviousConnID [a]	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
ThisDNRole	Mandatory
ThirdPartyDN	Optional
ThisQueue [b]	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. The attribute must appear if the value of `CallType` is `Consult`.
b. The attribute must appear in case of an ACD call.

Example



EventRinging Feature Example

EventEstablished

Description

For the application associated with the calling party: the telephony object specified by `OtherDN` has answered (either the calling party answered or the switch simulated an answer if option `auto-answer` is set on the switch) and the connection has been established. For the application associated with the called party: the call associated with `ConnID` has been established.

Contents

Event Attribute	Type
Event	Mandatory
Server	Mandatory
CustomerID	Optional
ReferenceID	Optional
ConnID	Mandatory
PreviousConnID [a]	Optional
CallID	Mandatory
CollectedDigits	Optional
CallHistory	Optional
CallType	Mandatory
CallState	Optional
AgentID	Optional
ThisDN	Mandatory
ThisQueue	Optional
ThisDNRole	Mandatory
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN	Optional
OtherTrunk	Optional
OtherQueue	Optional
OtherDNRole	Optional
DNIS	Optional
ANI	Optional
UserData	Optional
Reasons	Optional
Extensions	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional

a. The attribute must appear if the value of `CallType` is `Consult`.

EventAbandoned

Description

The caller abandoned the call before it was answered.

Contents

Event Attribute	Type
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallState	Mandatory
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
PreviousConnID [a]	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN	Optional
OtherDNRole	Optional
OtherQueue	Optional
OtherTrunk	Optional
Server	Mandatory
ThisDN	Mandatory
ThisDNRole	Mandatory
ThisQueue [b]	Optional
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. The attribute must appear if the value of `CallType` is `Consult`.
b. The attribute must appear in case of an ACD call.

EventDestinationBusy

Description

The called party specified by `OtherDN` is busy with another call.

Contents

Event Attribute	Type
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallState [a]	Optional
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN	Optional
OtherQueue	Optional
OtherTrunk	Optional
PreviousConnID [b]	Optional
Server	Mandatory
ThisDN	Mandatory
ThisDNRole	Mandatory
ThisQueue	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. For scenarios initiated with `RequestMakeCall`, this attribute may have values that clarify the reason for the destination being busy, for instance `CallStateSitInvalidNum`.
b. The attribute must appear if the value of `CallType` is `Consult`.

EventDiverted

Description

The call has been diverted from the queue to another telephony object.

Contents

Event Attribute	Type
CallHistory	Optional
CallID	Mandatory
CallState	Mandatory
CallType	Mandatory
ConnID	Mandatory
CollectedDigits	Optional
CustomerID	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN	Optional
OtherDNRole	Optional
OtherQueue	Optional
OtherTrunk	Optional
PreviousConnID [a]	Optional
Server	Mandatory
ThirdPartyDN [b]	Optional
ThirdPartyDNRole	Optional
ThirdPartyQueue [b]	Optional
ThisDN [c]	Mandatory
ThisDNRole	Mandatory
ThisQueuec	Mandatory
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. The attribute must appear if the value of `CallType` is `Consult`.
b. Attributes must be present if the value of `CallState` is `Redirected`. (See Redirect-Call Service.) In all other call scenarios, `ThirdPartyDN` must be present only if such information is provided by a CTI link.
c. These attributes must be equal.

EventHeld

Description

The call has been placed on hold.

Contents

Event Attribute	Type
AgentID	Optional
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN	Optional
OtherDNRole	Optional
OtherQueue	Optional
OtherTrunk	Optional
PreviousConnID [a]	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
ThisDNRole	Mandatory
ThisQueue	Optional
time	Mandatory

Event Attribute	Type
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. The attribute must appear if the value of `CallType` is `Consult`.

EventNetworkReached

Description

The call has reached the public network interface.

Contents

Event Attribute	Type
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN	Optional
OtherDNRole	Optional
OtherTrunk	Optional
PreviousConnID [a]	Optional
Server	Mandatory
ThisDN	Mandatory
ThisDNRole	Mandatory
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional

Event Attribute	Type
UserData	Optional

a. The attribute must appear if the value of `CallType` is `Consult`.

EventPartyAdded

Description

One or more parties has been added to the call as a result of a conference.

If only one party is added (as in the case of a simple conference call), the corresponding telephony object is specified in `OtherDN`.

If more than one party is added, then the corresponding telephony objects are specified in one of the following ways (depending on a particular T-Server):

- If a single EventPartyAdded message, the following keys are specified in the `Extensions` attribute:
  - `NumOfConsultDNs` key—The number of added parties.
  - `ConsultDN-`*k* keys—The directory number of the *k*-th added party.
- If multiple EventPartyAdded messages, one for each party on the consultation call joining the main call, the corresponding telephony object is specified in `OtherDN`.

Contents

Event Attribute	Type
AgentID	Optional
ANI	Optional
CallHistory	Optional
CallState	Optional
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN	Optional
OtherDNRole	Optional

Event Attribute	Type
OtherQueue	Optional
OtherTrunk	Optional
Server	Mandatory
ThirdPartyDN [a]	Mandatory
ThirdPartyDNRole [a]	Mandatory
ThisDN	Mandatory
ThisDNRole	Mandatory
ThisQueue	Optional
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. This attribute is not present if the switch does not distribute it to T-Server.

EventPartyChanged

Description

The telephony object specified by `OtherDN` has replaced the telephony object specified by `OtherDN` in the previously received event; or the `PreviousConnID` of the call has been given a new value, `ConnID`.

Contents

Event Attribute	Type
AgentID	Optional
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallState [a, d]	Mandatory
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory

Event Attribute	Type
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN [b]	Optional
OtherDNRole [b]	Optional
OtherTrunk [b]	Optional
PreviousConnID	Mandatory
Server	Mandatory
ThirdParty [c]	Mandatory
ThirdPartyDNRole [b]	Mandatory
ThisDN	Mandatory
ThisDNRole	Mandatory
ThisQueue	Optional
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. The value can be either Transferred or Conferenced. For more information, see Call Models and Flows.
b. The attribute must not appear if the `CallState` is `Conferenced`.
c. This attribute is not present if the switch does not distribute it to T-Server.
d. The value can be `OK` in SIP Server in multi-site transfers and conferences. For more information, see Special case: Multi-site ISCC Transfers and Conferences.

EventPartyDeleted

Description

The telephony object specified by `OtherDN` has been deleted from the conference call in question.

Contents

Event Attribute	Type
AgentID	Optional
ANI	Optional
CallHistory	Optional

Event Attribute	Type
CallID	Mandatory
CallState [a]	Mandatory
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN	Optional
OtherDNRole	Optional
OtherQueue	Optional
OtherTrunk	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThirdPartyDN	Optional
ThirdPartyDNRole	Optional
ThisDN	Mandatory
ThisDNRole	Mandatory
ThisQueue	Optional
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. The attribute indicates whether a call is still considered as a conference (that is, the number of parties in the call is more than two).

EventQueued

Description

The call has been queued in the ACD group specified by `ThisQueue`.

Contents

Event Attribute	Type
AgentID	Optional
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallState	Optional
CallType	Mandatory
CollectedDigits	Optional
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
LastCollectedDigit	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN	Optional
OtherDNRole	Optional
OtherQueue	Optional
OtherTrunk	Optional
PreviousConnID [a]	Optional
Server	Mandatory
ThisDN [b]	Mandatory
ThisDNRole	Mandatory
ThirdPartyDN	Optional
ThisQueueb	Mandatory
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. The attribute must appear if the value of `CallType` is `Consult`.
b. These attributes must be equal.

EventBridged

Description

Used in situations where Coverage Path is available or bridged calls can be handled. `EventBridged` indicates an extension, besides the one pointed to by `ThisDN`, has picked up the call, and that the telephony object specified by `ThisDN` is no longer ringing (although it still can pick up the call, establishing a three-way conversation).

`EventBridged` is used to describe a state where the call is neither ringing nor established. The nature of a bridge is such that it is possible for a call to move to a `bridged` state even after it has been released. Because of this, only calls released through CTI will be detected as moving into the `bridged` state. If a client issues a `RequestReleaseCall()` on a `bridged` call with two or more active bridged or bridging parties, `EventReleased`, with `CallState = CallStateBridged`, follows for the releasing party; the releasing party then receives `EventBridged`. At this point, a client may issue a `RequestAnswerCall()` to activate the call. If the client does this, `EventEstablished` follows.

Notes:

- Currently, T-Server does not fully support placing a bridged or bridging call on hold when there is only one active bridge. If a client attempts to place such a call on hold, T-Server does not reflect the held state of all members.
- Transferring and conferencing a bridged or bridging call currently works only when there is no more than one active bridge member.
- There may be cases where a call is made from a bridged DN to the principle extension on the bridge. In such an instance, the dialing and ringing between two DNs on the same bridge is happening within the context of a single call. While such a circumstance is supported, the bridged appearance of the call should be ignored and not used.

Contents

Event Attribute	Type
Event	Mandatory
Server	Mandatory
CustomerID	Optional
ConnID	Mandatory
PreviousConnID [a]	Optional
CallID	Mandatory
CollectedDigits	Optional
CallHistory	Optional
CallType	Mandatory
CallState [b]	Optional
AgentID	Optional
ThisDN	Mandatory

Event Attribute	Type
ThisQueue [c]	Optional
ThisDNRole	Mandatory
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN [b]	Optional
OtherTrunk [b]	Optional
OtherQueue	Optional
OtherDNRoleb	Optional
DNIS	Optional
ANI	Optional
UserData	Optional
Extensions	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional

a. The attribute must appear if the value of `CallType` is `Consult`.
b. For the Avaya Communication Manager only: In a Coverage Path scenario, the second party that has answered a call must receive `CallState=Conferenced`, but does not receive information about the other party (`OtherDN = NULL`). See Call Models and Flows.
c. The attribute must appear in case of an ACD call.

EventReleased

Description

The telephony object specified by `ThisDN` has disconnected or has been dropped from the call.

Contents

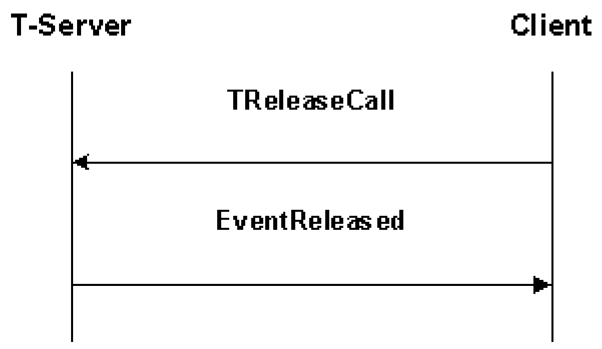
Event Attribute	Type
AgentID	Optional
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallState	Mandatory
Cause	Optional
CallType	Mandatory

Event Attribute	Type
ConnID	Mandatory
CollectedDigits	Optional
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN [a]	Optional
OtherDNRole [a]	Optional
OtherQueue [a]	Optional
OtherTrunk [a]	Optional
PreviousConnID [b]	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThirdPartyDN [c]	Optional
ThisDN	Mandatory
ThisDNRole	Mandatory
ThisQueue	Optional
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. This attribute does not appear if the release is from a conference. In all other call scenarios, the attribute must be present only if such information is provided by a CTI link.
b. The attribute must appear if the value of `CallType` is `Consult`.
c. The appearance of `ThirdPartyDN` depends on the following conditions:

- If information about the new destination is available from the switch at the moment when `EventReleased` is generated, then `ThirdPartyDN` is mandatory. Or, if T-Server has initiated a single-step transfer, redirection, or previously set the forwarding target, this attribute is also mandatory.
- If a call has gone through a single-step transfer, been redirected, or forwarded by another application (not the T-Server in question), this attribute is absent.

Example



EventReleased Feature Example

For more information, refer to Call Models and Flows.

EventRetrieved

Description

The call has been retrieved from hold.

Contents

Event Attribute	Type
AgentID	Optional
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallState	Mandatory
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
OtherDN [a]	Optional
OtherDNRole [a]	Optional
OtherQueue [a]	Optional

Event Attribute	Type
OtherTrunk [a]	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
ThisDNRole [b]	Mandatory
ThisQueue [c]	Optional
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. In all call scenarios, this attribute must be present only if the information is provided by a CTI link.
b. The value here is the same as that for the events preceding `EventRetrieved` (`EventEstablished` and `EventRinging`) for the same call.
c. The value here is the same as that for the events preceding `EventRetrieved` (`EventEstablished` and `EventRinging`) for the same call. (For non-ACD calls, `ThisQueue` is not reported.)

Network Attended Transfer Events

EventNetworkCallStatus

Description

Contains all pertinent information about the current state of a multi-party network call. It is sent to all interested parties whenever the call's state changes.

Contents

Event Attribute	Type
ANI	Optional
CallHistory	Optional
CallID	Mandatory
NetworkCallID	Optional
NetworkNodeID	Optional
CallType	Mandatory
ConnID	Mandatory
Cause	Optional
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
NetworkCallState	Mandatory
NetworkPartyRole [a]	Mandatory
NetworkOrigDN	Optional
NetworkDestDN	Optional
NetworkDestState	Mandatory
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
ThisDNRole	Mandatory
time	Mandatory
TransferredNetworkCallID	Optional

Event Attribute	Type
TransferredNetworkNodeID	Optional
UserData	Optional

a. This attribute is not present for events distributed by network T-Servers.

EventNetworkPrivateInfo

Description

This event is generated both in response to a `TNetworkPrivateService()` function call and when it is used to notify selected sets of clients of T-Server–specific information. This event indicates that one of two forms of data has been passed:

- Information supported only by certain T-Servers, and which is not covered by general feature requests.
- Notification about changes in T-Server behavior or device/call statuses.

This event can be distributed to the following clients:

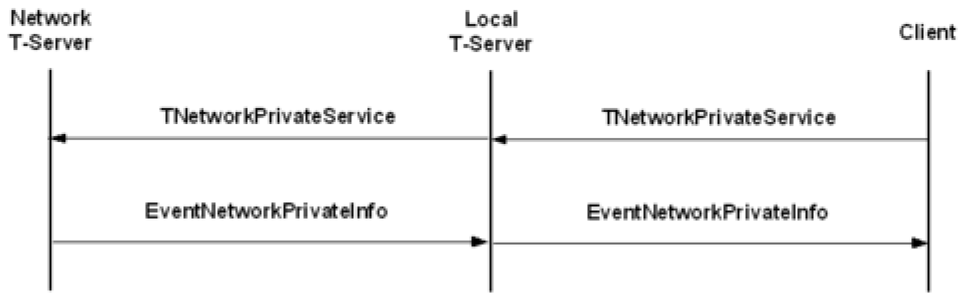
- Those who made a request for the private service, `TNetworkPrivateService()`.
- Those registered on the specified device, depending on the nature of the service.
- Those who would otherwise be ineligible (for security reasons) for receiving given events, and who thus require additional registration in order to be notified.

Contents

Event Attribute	Type
PrivateEvent	Mandatory
Event	Mandatory
Reasons	Optional
UserData	Optional
Extensions	Optional
ReferenceID	Optional
Server	Mandatory
ConnID	Optional
ThisDN	Optional
NetworkCallState	Optional
NetworkPartyRole	Optional
NetworkOrigDN	Optional
NetworkDestDN	Optional

Event Attribute	Type
NetworkDestState	Optional
time	Mandatory

Example



EventNetworkPrivateInfo Example

Call-Routing Events

EventRouteRequest

Description

The call has been placed on the routing point specified by `ThisDN`, and the switch is waiting for routing instructions.

Contents

Event Attribute	Type
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallType	Mandatory
CollectedDigits	Optional
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
LastCollectedDigit	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN	Optional
OtherDNRole	Optional
OtherQueue	Optional
OtherTrunk	Optional
PreviousConnID [a]	Optional
Server	Mandatory
ThisDN [b]	Mandatory
ThisQueue [b]	Mandatory
ThisTrunk	Optional
ThirdPartyDN	Optional
time	Mandatory

Event Attribute	Type
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. The `PreviousConnID` attribute must appear if a call with `CallType=Consult` has been placed on a routing point.
b. `ThisDN` and `ThisQueue` attributes must have equal values.

Examples

See EventRegistered and the figure EventRouteRequest and EventRouteUsed Feature, Example 1.

EventRouteUsed

Description

The call has been routed as requested in the function `TRouteCall()` or has been default routed by the switch after the routing timeout has expired (that is, there was no routing instruction from the computer domain within the specified timeout).

Contents

Event Attribute	Type
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallState	Optional
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThirdPartyDN [a]	Optional

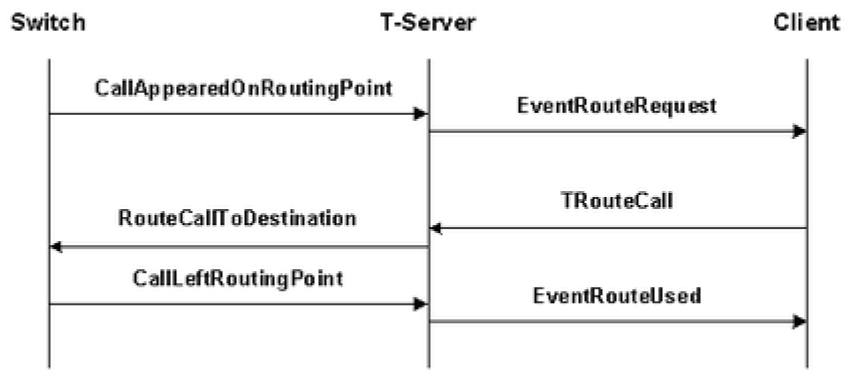
Event Attribute	Type
OtherDN [b]	Optional
ThirdPartyDNRole = Destination	Optional
ThisDN [c]	Mandatory
ThisQueue [c]	Mandatory
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. This attribute specifies the destination DN or dialing number. It is:

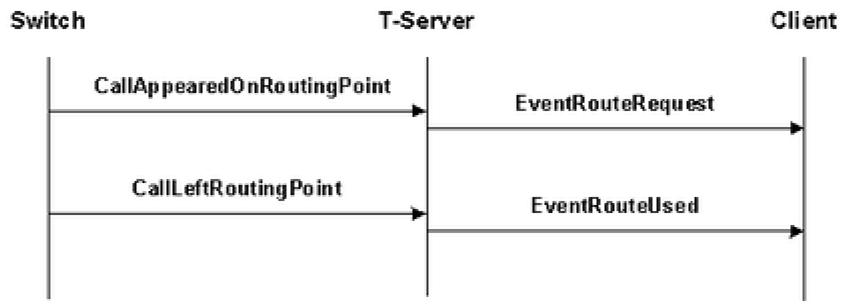
- Mandatory if routing was done by URS (or another routing application connected to T-Server).
- Absent if the call was rejected. Optional in other cases.

b. The OtherDN attribute is used to specify the target party when the forward feature is in progress.
c. These attributes must be equal.

Example



EventRouteRequest and EventRouteUsed Feature, Example 1



EventRouteRequest and EventRouteUsed Feature, Example 2 (if the Routing Timeout Expires Before the TRouteCall Request Generated by Computer Domain [Interaction Router])

Call-Treatment Events

EventTreatmentApplied

Description

The call has been treated, and the Treatment Device (TD) is processing the treatment instruction.

Contents

Event Attribute	Type
CallID	Mandatory
CallType	Mandatory
ConnID	Mandatory
CustomerID	Mandatory
Event	Mandatory
Extensions	Mandatory
NetworkCallID	Optional
NetworkNodeID	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
ThisDNRole	Mandatory
time	Mandatory
TransferConnID	Optional
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
TreatmentParameters	Optional
TreatmentType	Mandatory
UserData	Optional

EventTreatmentEnd

Description

The call has been treated and the Treatment Device (TD) is waiting for another instruction.

Important

This event does not appear in cases of continuing treatments like `Silence` or `RingBack`.

Contents

Event Attribute	Type
CallID	Mandatory
CallType	Mandatory
CollectedDigits [a]	Optional
ConnID	Mandatory
CustomerID	Mandatory
Event	Mandatory
Extensions [b]	Optional
LastCollectedDigit [a]	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
time	Mandatory
TransferConnID	Optional
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
TreatmentParameters	Optional
TreatmentType	Mandatory
UserData	Optional

a. The attributes are present if `TreatmentType` is either `CollectDigits` or `PlayAnnouncementAndCollectDigits`.
b. The following key-value pairs are set for all Treatment Types, except in the case of the Nortel

Communication Server 2000/2100:

- For all Treatment Types where an announcement was played, `INTERRUPTED` is set to:
  - `NO`, if the announcement was not interrupted.
  - `KEYPAD`, if it was interrupted by keypad entry.
  - `VOICE`, if it was interrupted by the caller speaking something.
- For all Treatment Types where digits are to be collected from the caller, `COMPLETION_STATUS` is set to:
  - `NORMAL`, if the treatment completed normally (optional).
  - `TIMEOUT`, if the digit collection timed out before all required digits could be collected.
  - `CANCELLED`, if the treatment was cancelled by a request from router.
- For TreatmentType=DigitsVerification only, the following key-value pairs apply:
  - `VERIFICATION_STATUS` (the result of digits verification) is set to 1 if verification succeed, 0 if it did not.
  - `ATTEMPTS` is set to the number of digit-collection attempts made.
- For `TreatmentType=RecordUserAnnouncement`, the following key-value pair applies:
  - `USER_ANN_ID` is set to the message identifier, an integer, recorded by the user specified with `USER_ID`.

EventTreatmentNotApplied

Description

The call has not been treated for some reason. The reason is returned in `ErrorCode` and `ErrorMessage` parameters.

Contents

Event Attribute	Type
CallID	Mandatory
CallType	Mandatory
ConnID	Mandatory
CustomerID	Mandatory
ErrorCode	Mandatory
ErrorMessage	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional

Event Attribute	Type
Reasons	Optional
ReferenceID	Mandatory
Server	Mandatory
ThisDN	Mandatory
time	Mandatory
TransferConnID	Optional
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
TreatmentParameters	Optional
TreatmentType	Mandatory
UserData	Optional

EventTreatmentRequired

Description

A call has been placed to a treatment device port specified by `ThisDN`, and the switch or the treatment device is waiting for treatment instructions.

Contents

Event Attribute	Type
ANI	Optional
CallID	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Mandatory
Event	Mandatory
Extensions	Optional
Server	Mandatory
ThisDN	Mandatory
time	Mandatory
UserData	Optional

DTMF Events

EventDigitsCollected

Description

The digits specified by `CollectedDigits` have been collected. This event is sent either to the DN representing the device collecting the digits or to all monitored parties for the call.

Contents

Event Attribute	Type
CallHistory	Optional
CallID	Mandatory
CallState	Mandatory
CallType	Mandatory
CollectedDigits	Mandatory
ConnID	Mandatory
CustomerID	Optional
Event	Mandatory
Extensions	Optional
LastCollectedDigit	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN [a]	Optional
OtherDNRole	Optional
OtherQueue	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
ThisDNRole	Mandatory
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional

Event Attribute	Type
UserData	Optional

a. This attribute indicates the source of the collected digits, if T-Server can identify that source.

EventDTMFSent

Description

The specified DTMF sequence has been sent.

Contents

Event Attribute	Type
ANI	Optional
CallHistory	Optional
CallID	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

Voicemail Events

EventMailBoxLogin

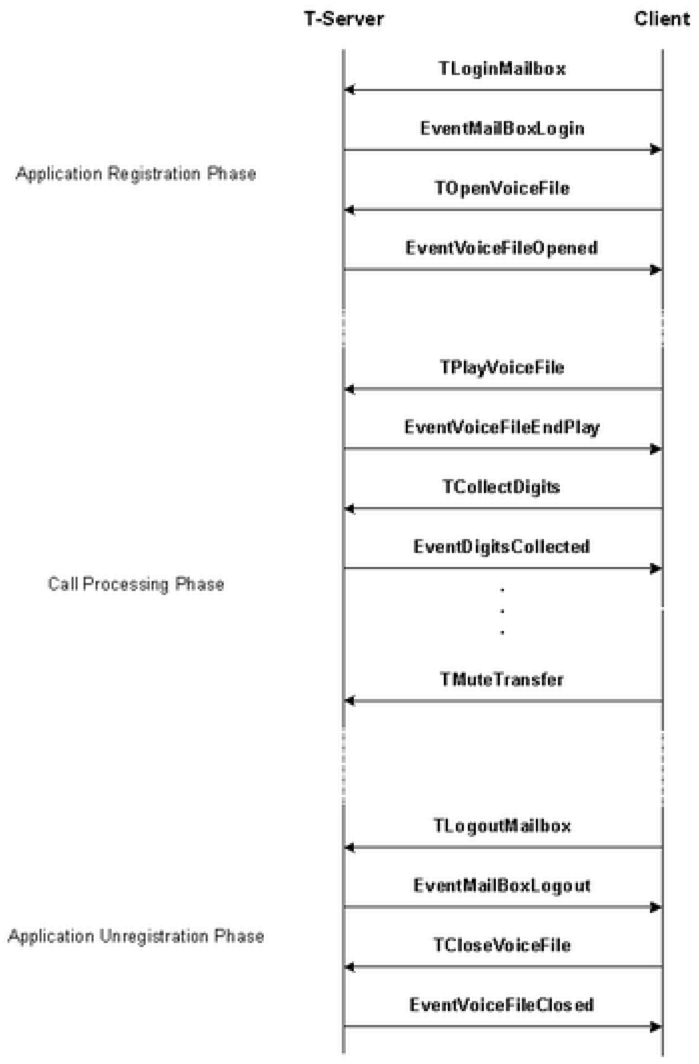
Description

The telephony object specified by `ThisDN` has logged in to the mailbox specified by the `mbox_number` parameter in the corresponding `TLoginMailBox()` function.

Contents

Event Attribute	Type
CustomerID	Optional
Event	Mandatory
Reasons	Optional
ReferenceID	Mandatory
Server	Mandatory
ThisDN	Mandatory
time	Mandatory
Extensions	Optional

Example



Voice-Processing Feature Example

EventMailBoxLogout

Description

The telephony object specified by `ThisDN` has logged out of the mailbox it logged in to earlier.

Contents

Event Attribute	Type
CustomerID	Optional
Event	Mandatory
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
time	Mandatory
Extensions	Optional

Example

See Voice-Processing Feature Example.

EventVoiceFileOpened

Description

The voice file specified by the `file_handle` parameter in the corresponding `TOpenVoiceFile()` function has been opened.

Contents

Event Attribute	Type
CustomerID	Optional
Event	Mandatory
Extensions	Optional
FileHandle	Mandatory
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
time	Mandatory
UserData	Optional

Example

See Voice-Processing Feature Example.

EventVoiceFileClosed

Description

The voice file specified by the `file_handle` parameter in the corresponding `TCloseVoiceFile()` function has been closed.

Contents

Event Attribute	Type
CustomerID	Optional
Event	Mandatory
Extensions	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
time	Mandatory
UserData	Optional

Example

See Voice-Processing Feature Example.

EventVoiceFileEndPlay

Description

The voice file specified by the `file_handle` parameter in the corresponding `TPlayVoice()` function has been played back completely.

Contents

Event Attribute	Type
CallID	Mandatory

Event Attribute	Type
CallState	Mandatory
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
Event	Mandatory
Extensions	Optional
FileHandle	Mandatory
NetworkCallID	Optional
NetworkNodeID	Optional
Reasons	Optional
ReferenceID	Mandatory
Server	Mandatory
ThisDN	Mandatory
ThisDNRole	Mandatory
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

Example

See Voice-Processing Feature Example.

Agent-State and DN Events

EventAgentLogin

Description

The agent has logged in to the ACD group specified by `ThisQueue`.

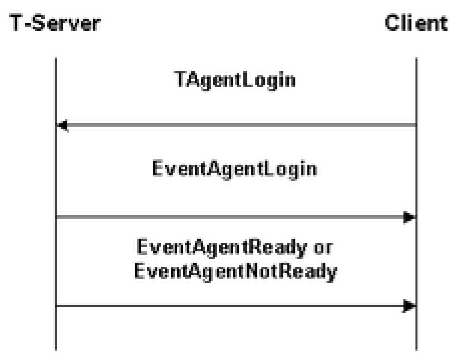
> **Important**
> Multiple agent logins are allowed for the same DN and agent ID combination (since `EventAgentLogin` does not indicate by itself a transition of agent state).

Contents

Event Attribute	Type
AgentID [a]	Optional
CustomerID	Optional
Event	Mandatory
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
ThisQueue	Optional
time	Mandatory
WorkMode	Optional
Extensions [b]	Optional

a. `AgentID` must be present if the agent is logged in through T-Server or if the information is available.
b. If present, the `Extensions` attribute may include a `ReasonCode` value specifically used to communicate hardware reasons.

Example



EventAgentLogin Feature Example

EventAgentLogout

Description

The agent has logged out of the ACD group specified by `ThisQueue`.

Important
With CTI platforms that support agent login for multiple queues, this event signals that the agent has been moved to the `Logged Out` state, and is thus used only for an agent's final logout. (`EventQueueLogout`, on the other hand, indicates that an agent remains logged in to some other ACD queue. See EventQueueLogout.)

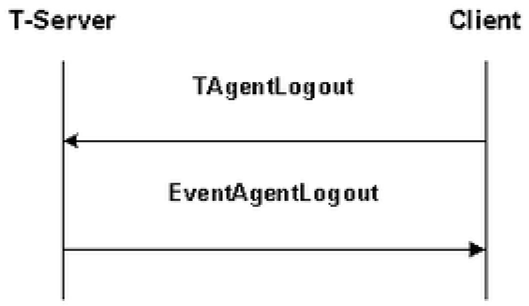
Contents

Event Attribute	Type
Event	Mandatory
Server	Mandatory
ReferenceID	Optional
CustomerID	Optional
AgentID [a]	Optional
ThisDN	Mandatory
ThisQueue	Optional
Reasons	Optional
time	Mandatory

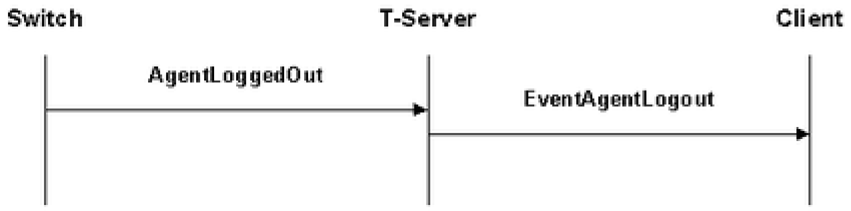
Event Attribute	Type
Extensions [b]	Optional

a. `AgentID` must be present if the agent is logged in through T-Server or if the information is available.
b. If present, the `Extensions` attribute may include a `ReasonCode` value specifically used to communicate hardware reasons.

Examples



EventAgentLogout (Through Link) Feature, Example 1



EventAgentLogout (Through PhoneSet) Feature, Example 2

EventQueueLogout

Description

The agent has logged out of the ACD queue specified by `ThisQueue`, but remains logged in to some other ACD queue.

Contents

Event Attribute	Type
Event	Mandatory

Event Attribute	Type
Server	Mandatory
ReferenceID	Optional
CustomerID	Optional
AgentID [a]	Optional
ThisDN	Mandatory
ThisQueue	Optional
Reasons	Optional
time	Mandatory
Extensions [b]	Optional

a. `AgentID` must be present if the agent is logged in through T-Server or if the information is available.
b. If present, the `Extensions` attribute may include a `ReasonCode` value specifically used to communicate hardware reasons.

EventAgentReady

Description

The agent is ready to receive ACD calls.

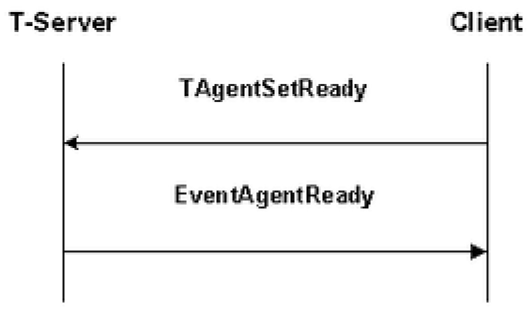
Contents

Event Attribute	Type
AgentID [a]	Optional
CustomerID	Optional
Event	Mandatory
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
ThisQueue	Optional
time	Mandatory
WorkMode [b]	Mandatory
Extensions [c]	Optional

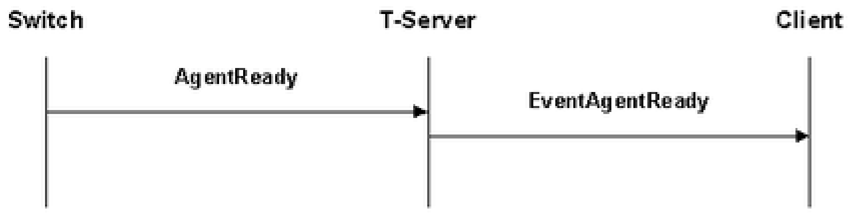
a. `AgentID` must be present if the agent is logged in through T-Server or if the information is available.

b. `WorkMode` is mandatory for the Avaya Communication Manager T-Server when not in Soft Agent mode.
c. If present, the `Extensions` attribute may include a `ReasonCode` value specifically used to communicate hardware reasons.

Examples



EventAgentReady (Through Link) Feature, Example 1



EventAgentReady (Through PhoneSet) Feature, Example 2

EventAgentNotReady

Description

The agent is not ready to receive ACD calls.

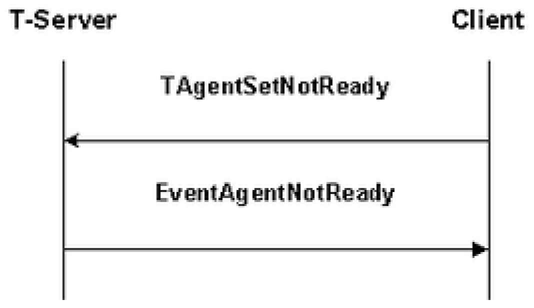
Contents

Event Attribute	Type
AgentID [a]	Optional
CustomerID	Optional
Event	Mandatory
Reasons	Optional

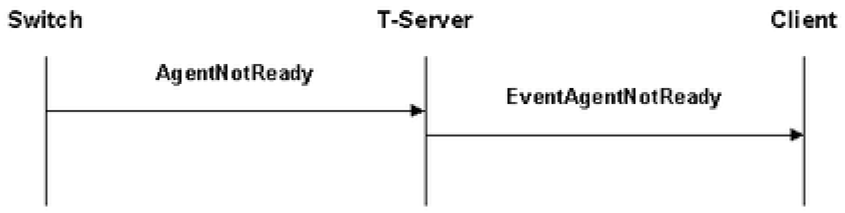
Event Attribute	Type
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
ThisQueue	Optional
time	Mandatory
WorkMode [b]	Optional
Extensions [c]	Optional

a. `AgentID` must be present if the agent is logged in through T-Server or if the information is available.
b. `WorkMode` is mandatory for the Avaya Communication Manager T-Server when not in Soft Agent mode.
c. If present, the `Extensions` attribute may include a `ReasonCode` value specifically used to communicate hardware reasons.

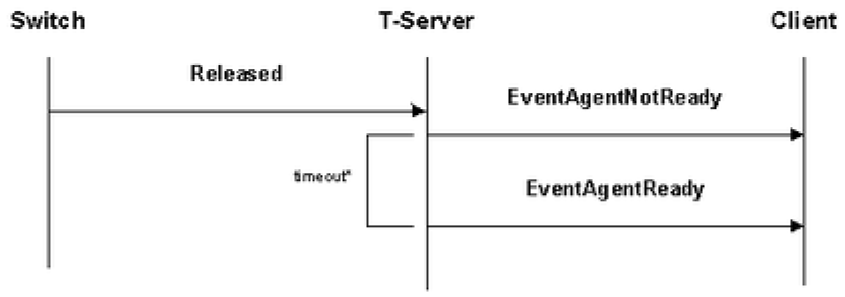
Examples



EventAgentNotReady (Through Link) Feature, Example 1



EventAgentNotReady (Through PhoneSet) Feature, Example 2



EventAgentNotReady Feature, Example 3 (for the Nortel Communication Server 2000/2100 only)

Important
A timeout in EventAgentNotReady Feature, Example 3 (for the Nortel Communication Server 2000/2100 (*) is specified as the wrap-up time in the Configuration Layer. T-Server distributes `EventAgentReady` automatically if there is no activity on the DN that can be considered an agent-state change within the specified timeout. See also EventAgentLogin.

EventAgentAfterCallWork (Obsolete — No Longer Supported)

Description

The agent is performing administrative duties for a previous call (that is, updating some information) and, therefore, is not ready to receive further ACD calls.

EventAgentIdleReasonSet (Obsolete—No Longer Supported)

Description

Indicates that the Idle reason for the telephony object specified by the parameter `ThisDN` has been successfully set.

Contents

Event Attribute	Type
AgentID	Optional

Event Attribute	Type
CustomerID	Optional
Event	Mandatory
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
ThisQueue	Optional
time	Mandatory
Extensions	Optional

EventDNOutOfService

Description

The DN specified in the `ThisDN` attribute is out of service and cannot make or receive calls. This event is generated when an out-of-service state is first detected or when a new client registers on a DN known to be out of service.

When a DN is out of service, only the following T-Library requests can be issued for it: client registration and unregistration, queries, agent login, and private service requests.

> **Important**
> T-Server returns a `TERR_OUT_OF_SERVICE` error if it is called on to attempt a supported operation that cannot progress on an out-of-service DN.

When a DN goes out of service, T-Server notifies the user about the termination of active calls or changes an agent state (not ready/logout) using normal T-Library events. The other applications should rely only on those events to change the DN/agent state.

Contents

Event Attribute	Type
ThisDN	Mandatory
Extensions	Optional

Example

See EventDNBackInService.

EventDNBackInService

Description

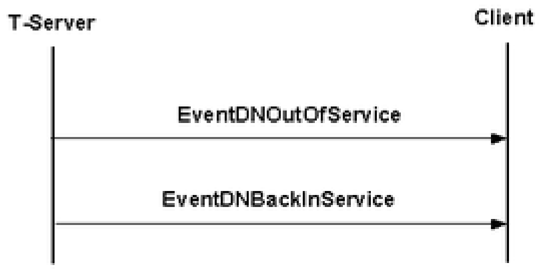
The DN specified in the ThisDN attribute is back in service and can make or receive calls. This event is generated when a DN, which has been out of service and for which the EventDNOutOfService was previously distributed, returns to service.

In the absence of EventDNOutOfService and EventDNBackInService, all clients should assume, for backward-compatibility reasons, that the DN is in service.

Contents

Event Attribute	Type
ThisDN	Mandatory
Extensions	Optional

Example



EventDNBackInService Feature Example

Important
Between EventDNOutOfService and EventDNBackInService, the client is not able to perform any requests, and no events should be expected during this outage. Genesys recommends that you perform TQueryAddress() after EventDNBackInService to ensure synchronization between T-Server and client.

EventDNDOn

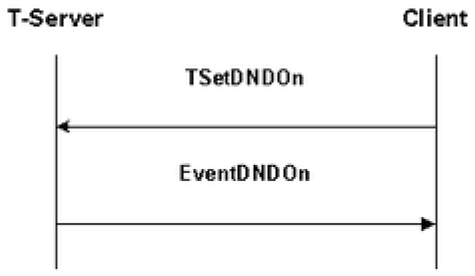
Description

The Do-Not-Disturb (DND) feature has been turned on for the telephony object specified by ThisDN.

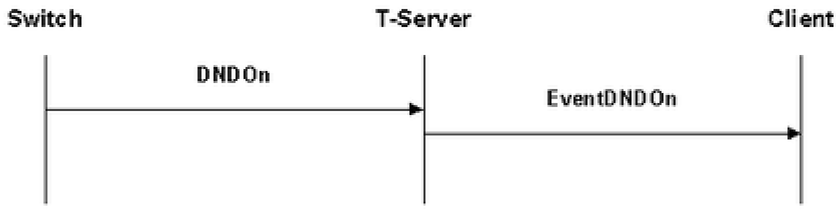
Contents

Event Attribute	Type
CustomerID	Optional
Event	Mandatory
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
time	Mandatory
Extensions	Optional

Examples



EventDNDOn (Through Link) Feature, Example 1



EventDNDOn (Through PhoneSet) Feature, Example 2

EventDNDOff

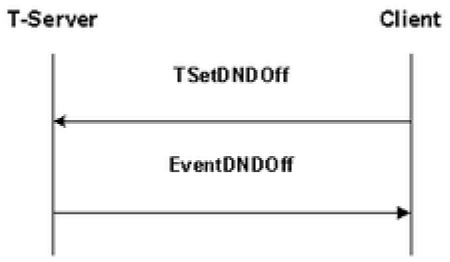
Description

The Do-Not-Disturb (DND) feature has been turned off for the telephony object specified by `ThisDN`.

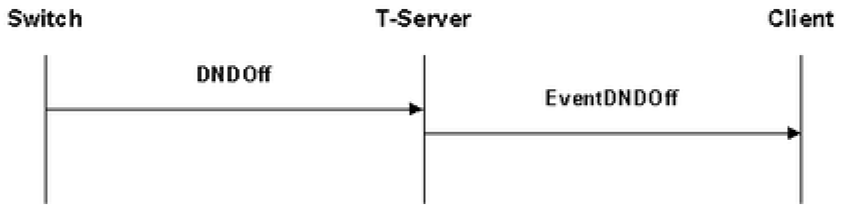
Contents

Event Attribute	Type
CustomerID	Optional
Event	Mandatory
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
time	Mandatory
Extensions	Optional

Examples



EventDNDOff (Through PhoneSet) Feature, Example 1



EventDNDOff (Through PhoneSet) Feature, Example 2

EventForwardSet

Description

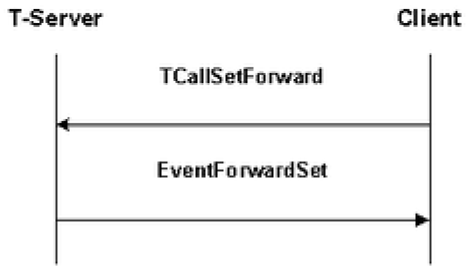
The Forwarding feature has been turned on for the telephony object specified by `ThisDN`.

Contents

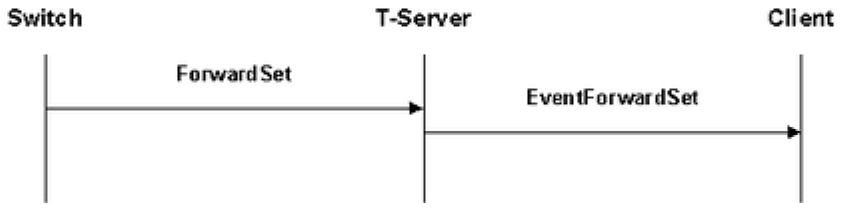
Event Attribute	Type
CustomerID	Optional
Event	Mandatory
Extensions	Optional
InfoStatus (CallForwardingStatus, SendAllCallsStatus)	Optional
OtherDN [a]	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
time	Mandatory
ForwardMode	Optional

a. The `OtherDN` attribute is used to specify the target party when the Forward feature is in progress.

Examples



EventForwardSet (Through Link) Feature, Example 1



EventForwardSet (Through PhoneSet) Feature, Example 2

EventForwardCancel

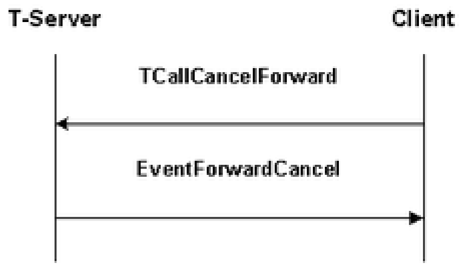
Description

The Forwarding feature has been turned off for the telephony object specified by `ThisDN`.

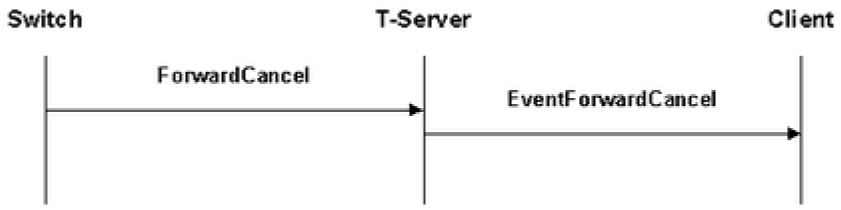
Contents

Event Attribute	Type
CustomerID	Optional
Event	Mandatory
Extensions	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
time	Mandatory

Examples



EventForwardCancel (Through Link) Feature, Example 1



EventForwardCancel (Through PhoneSet) Feature, Example 2

EventMonitoringNextCall

Description

A request to monitor the next call(s) has been accepted. The event is delivered to the applications on the supervisor's and agent's desktops.

Important

A supervisor who monitors calls as a result of the request `TMonitorNextCall()` is able to hear and participate in conversations on the monitored DN. If it is necessary to receive events associated with conversations, the supervisor's soft phone must be registered with the various DNs that might be monitored.

Contents

Event Attribute	Type
ThisDN [a]	Mandatory
ThisDNRole [a]	Mandatory
OtherDN [b]	Mandatory
OtherDNRole [b]	Mandatory
ReferenceID	Optional
MonitorNextCallType	Mandatory
Reasons	Optional
Extensions	Optional

a. When the event is delivered to an application on the supervisor’s desktop, `ThisDN` is set to the supervisor’s DN and `ThisDNRole` is set to `DNRoleObserver`. When the event is delivered to an application on the agent’s desktop, `ThisDN` is set to the agent’s DN and `ThisDNRole` is set to `DNRoleDestination`.
b. When the event is delivered to an application on the supervisor’s desktop, `OtherDN` is set to the agent’s DN and `OtherDNRole` is set to `DNRoleDestination`. When the event is delivered to an application on the agent’s desktop, `OtherDN` is set to the supervisor’s DN and `OtherDNRole` is set to `DNRoleObserver`.

Example

See EventMonitoringCancelled.

EventMonitoringCancelled

Description

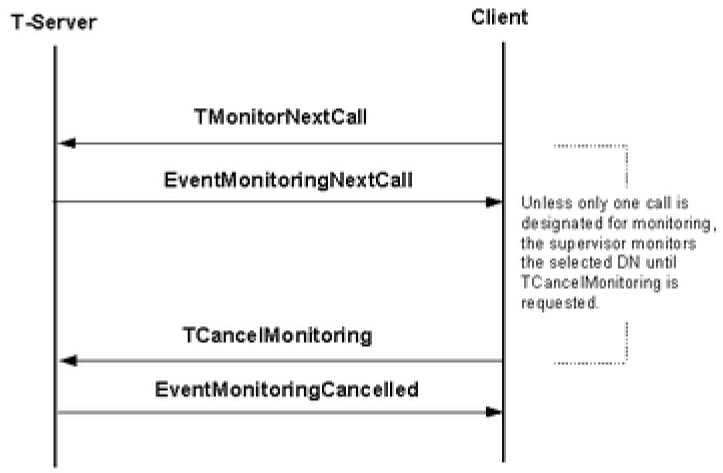
The call monitoring has been canceled either by a separate call to the `TMonitorNextCall()` function or to the `TCancelMonitoring()` function. The event is delivered to the applications on the supervisor's and agent's desktops.

Contents

Event Attribute	Type
ThisDN [a]	Mandatory
ThisDNRole [a]	Mandatory
OtherDN [b]	Mandatory
OtherDNRole [b]	Mandatory
ReferenceID	Optional
Reasons	Optional
Extensions	Optional

a. When the event is delivered to an application on the supervisor's desktop, `ThisDN` is set to the supervisor's DN and `ThisDNRole` is set to `DNRoleObserver`. When the event is delivered to an application on the agent's desktop, `ThisDN` is set to the agent's DN and `ThisDNRole` is set to `DNRoleDestination`.
b. When the event is delivered to an application on the supervisor's desktop, `OtherDN` is set to the agent's DN and `OtherDNRole` is set to `DNRoleDestination`. When the event is delivered to an application on the agent's desktop, `OtherDN` is set to the supervisor's DN and `OtherDNRole` is set to `DNRoleObserver`.

Example


T-Server
Client
TMonitorNextCall
EventMonitoringNextCall
Unless only one call is designated for monitoring, the supervisor monitors the selected DN until TCancelMonitoring is requested.
TCancelMonitoring
EventMonitoringCancelled

EventMonitoringCancelled Feature Example

EventOffHook

Description

The telephony object specified by `ThisDN` has gone off-hook.

Contents

Event Attribute	Type
AgentID	Optional
CallHistory	Optional
CallID	Optional
CallState	Optional
CallType	Optional
ConnID	Optional
CustomerID	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
Server	Mandatory
ThisDN	Mandatory

Event Attribute	Type
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

EventOnHook

Description

The telephony object specified by `ThisDN` has gone on-hook.

Contents

Event Attribute	Type
AgentID	Optional
CallHistory	Optional
CallID	Optional
ConnID	Optional
CustomerID	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
Server	Mandatory
ThisDN	Mandatory
ThisDNRole	Optional
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

EventMuteOn

Description

A party identified by `ThisDN` is now in the Mute mode.

Contents

Event Attribute	Type
ConnID	Mandatory
CustomerID	Optional
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
Reasons	Optional
ReferenceID	Optional
ThisDN	Mandatory
ThisDNRole	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

EventMuteOff

Description

A party identified by `ThisDN` is no longer in `Mute` (microphone-disabled) mode. The `ReferenceID` attribute is set to indicate the corresponding `TSetMuteOff()` function.

Contents

Event Attribute	Type
ConnID	Mandatory
CustomerID	Optional
Reasons	Optional
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional

Event Attribute	Type
ReferenceID	Optional
ThisDN	Mandatory
ThisDNRole	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

EventListenDisconnected

Description

The switch has registered `Deaf` mode for the specified telephony object (in `OtherDN`).

Contents

Event Attribute	Type
CallHistory	Optional
CallID	Mandatory
CallState [a]	Mandatory
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN [b]	Mandatory
OtherDNRole	Optional
OtherQueue	Optional
OtherTrunk	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThirdPartyDN [c]	Mandatory
ThisDN [d]	Mandatory
ThisDNRole	Mandatory

Event Attribute	Type
ThisQueue	Optional
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. The following `CallStates` are used:

- `CallStateOk`: the party can still participate in conversation with some active members of the conference.
- `CallStateDeafened`: the party cannot listen to the conversation, but can be heard by the conference members.
- `CallStateHeld`: the party cannot hear or be heard by the conference members.

b. Applies to the disconnected party.
c. The party that cannot be heard by the disconnected party.
d. The party that initiated the request.

EventListenReconnected

Description

The switch has canceled `Deaf` mode for the specified telephony object.

Contents

Event Attribute	Type
CallHistory	Optional
CallID	Mandatory
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN [a]	Mandatory
OtherDNRole	Optional

Event Attribute	Type
OtherQueue	Optional
OtherTrunk	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThirdPartyDN [b]	Optional
ThisDN [c]	Mandatory
ThisDNRole	Mandatory
ThisQueue	Optional
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. The reconnected party.
b. The party that is heard by the reconnected party.
c. The party that initiated the request.

EventMessageWaitingOn

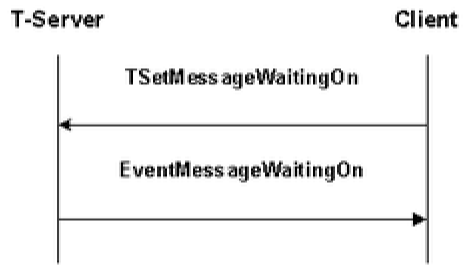
Description

The `Waiting` indicator has been turned on for the telephony object specified by `ThisDN`.

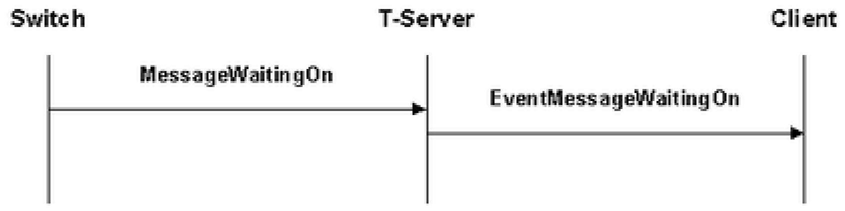
Contents

Event Attribute	Type
CustomerID	Optional
Event	Mandatory
Extensions	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
time	Mandatory

Examples



EventMessageWaitingOn (Through Link) Feature, Example 1



EventMessageWaitingOn (Through PhoneSet) Feature, Example 2

EventMessageWaitingOff

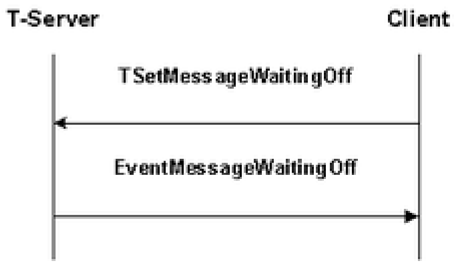
Description

The `Waiting` indicator has been turned off for the telephony object specified by `ThisDN`.

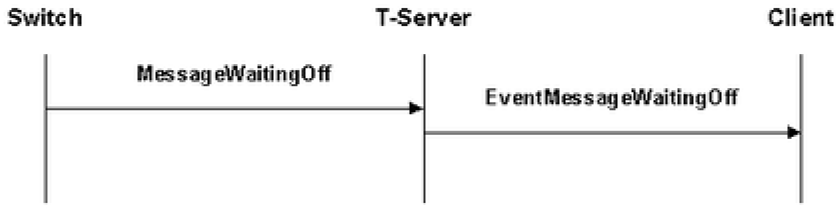
Contents

Event Attribute	Type
CustomerID	Optional
Event	Mandatory
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN	Mandatory
time	Mandatory
Extensions	Optional

Examples



EventMessageWaitingOff (Through Link) Feature,
Example 1



EventMessageWaitingOff (Through PhoneSet) Feature, Example 2

Query Events

EventAddressInfo

Description

This event is generated as a response to the `TQueryAddress()` function and includes information specified by `InfoType` and `InfoStatus` about the telephony object specified by either `ThisDN` or `ThisQueue`.

Contents

Event Attribute	Type
AgentID [a]	Optional
CallID	Optional
ConnID [b]	Optional
CustomerID	Optional
Event	Mandatory
Extensions [c]	Optional
InfoStatus	Mandatory
InfoType [c]	Mandatory
NetworkCallID	Optional
NetworkNodeID	Optional
PreviousConnID [b]	Optional
ReferenceID	Mandatory
Reasons [d]	Optional
Server	Mandatory
ThisDN	Mandatory
ThisQueue	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. The `AgentID` attribute can only be present for objects of `AddressTypePosition` or `AddressTypeDN`.
b. The `ConnID` and `PreviousConnID` attributes are mandatory when the `InfoType` parameter is set to

`AddressInfoCallsQuery`. `ConnID` is the connection ID for the active call; `PreviousConnID` is the connection ID for the held call, if any.
c. Based on the `InfoType` value, additional information is provided in the `InfoStatus`, which is a union element, and in `Extensions` attributes. See Table InfoStatus and Extensions in EventAddressInfo Available for All T-Servers for `InfoType=AddressInfoQueueStatus` and `AddressInfoDNStatus`, which are available on all T-Servers. See Table InfoStatus and Extensions in EventAddressInfo Available for Particular T-Servers for all other `InfoTypes` that are available on a device-specific basis. If a particular `InfoType` is not supported, T-Server responds with `EventError`.
d. If present, the value here is the last known KVList data supplied by recent activity for this address.

InfoStatus and Extensions in EventAddressInfo Available for All T-Servers

Attribute InfoStatus	**Attribute Extensions**		
Key	**Value Type**	**Value Description**	
InfoType=AddressInfoQueueStatus			
NumberOfCalls[a]	conn-%d	string	The connection ID of a call (converted into a string).
	ct-%d	integer	The CallType for the corresponding ConnectionID.
	mt-%d	integer	The media type of the corresponding ConnectionID. (The value may be absent here if the media type is voice. [b]) `0` (or absent) if voice `>0` for non-voice media
	ps-%d	integer	The Call-Party state for the corresponding Connection ID. See Unified Call-Party States for more details.
	status	integer	The DN status. [c]
InfoType=AddressInfoDNStatus			
NumberOfCalls[a]	conn-%d	string	The connection ID of a call (converted into a string).
	ct-%d	integer	The CallType for the corresponding connection ID.
	mt-%d	integer	The media type of the corresponding connection ID. (The

Attribute InfoStatus	Attribute Extensions		
			value may be absent here if the media type is voice. [b]) `0` (or absent) if voice `>0` for non-voice media
	ps-%d	integer	The Call-Party state for the corresponding Connection ID. See Unified Call-Party States for more details.
	queue-%d	string	The enumerated queues associated with an agent logged into a DN.
	AgentStatus	integer	The Agent status. [d]
	status	integer	The DN status. [c]
	dnd	integer	Do Not Disturb status. [e] Not present if T-Server has no information about the DND status.
	fwd	string	Specifies forwarding targets based on corresponding conditions. [f] Not present if T-Server has no information about Forward status.
	mwl	integer	Message Waiting Lamp status. [g] Not present if T-Server has no information about MWL status.

a. The information is based on what T-Server can provide; it may be inaccurate when T-Server is not in sync with the switch. It is possible for T-Server to return a non-error message with missing parameters in response to a request regarding an incorrectly configured DN or queried DN.
b. Client applications should set this value to `0` (zero) if it is absent in the event. This insures that the media type is understood to be voice.
c. The following are the DN statuses:

- `<0` (`UNKNOWN`): there is no available information about the DN status.
- `0` (`IDLE`): there is no activity on the DN.
- `>0` (`NOT_IDLE`): there is some activity on the DN—there is at least one call on the DN or the device is off-hook. Note the following special values:

- `0x80: DN is out of service.`
- `0xC0`: DN is undergoing maintenance.
- `0x90`: Device associated with DN is locked out.
- `0x88`: DN is vacant.

d. The following are the Agent statuses:

- `<0` (`UNKNOWN`): the status of an agent is unknown.
- `0` (`LOGGED_OUT`): there is no agent logged on this DN.
- 1 (`LOGGED_IN`): an agent is logged in but work mode is unknown.
- 2 (READY): the status of agent is Ready.
- 3 (`NOT_READY`): the status of agent is `NotReady`.
- 4 (`ACW`): the status of agent is `AfterCallWork`.
- 5 (`WALK_AWAY`): the status of agent is `WalkAway`.

Note: The `AgentStatus Extensions` key is delivered only for objects with an `Address` type of `Address`TypePosition or `AddressTypeDN`.

e. The following are the DND statuses:

- `0`: DND off
- `1`: DND on

f. Possible values are:

- `off`, on, or the designated destination DN (for unconditional forwarding), or the designated destination–DN list (with conditions for forwarding). The following conditions may be used: `OnBusy, OnNoAnswer, OnBusyAndNoAnswer, SendAllCalls.`
  - Examples:
    1. Unconditional Destination DN: `3000`
    2. Conditions with Destination-DN list: `OnBusy:1000 OnNoAnswer:2000`. (This forwards calls to two different DNs based on certain conditions being met.)

g. The following are the MWL statuses:

- `0`: MWL off
- `1`: MWL on

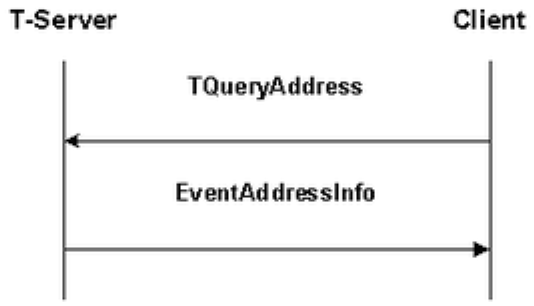
InfoStatus and Extensions in EventAddressInfo Available for Particular T-Servers

Attribute InfoStatus	Attribute Extensions		
Key	**Value Type**	**Value Description**	

Attribute InfoStatus	Attribute Extensions		
InfoType=AddressInfoAddressStatus			
AddressStatus			
InfoType=AddressInfoMessageWaitingStatus			
MsgWaitingStatus			
InfoType=AddressInfoAssociationStatus			
AssociationStatus			
InfoType=AddressInfoCallForwardingStatus			
CallForwardingStatus			
InfoType=AddressInfoAgentStatus			
AgentStatus			
InfoType=AddressInfoNumberOfAgentsInQueue			
NumberOfAgentsInQueue	AgentsInQueue	integer	Requested number is returned in AddressInfoStatus attribute; in addition, the Extensions attribute contains all of three keys
	AvailableAgents	integer	
	CallsInQueue	integer	
InfoType=AddressInfoNumberOfAvailableAgentsInQueue			
NumberOfAvailable-AgentsInQueue	AgentsInQueue	integer	Requested number is returned in AddressInfoStatus attribute; in addition, the Extensions attribute contains all of three keys
	AvailableAgents	integer	
	CallsInQueue	integer	
InfoType=AddressInfoNumberOfCallsInQueue			
NumberOfCallsInQueue	AgentsInQueue	integer	Requested number is returned in AddressInfoStatus attribute; in addition, the Extensions attribute contains all of three keys
	AvailableAgents	integer	
	CallsInQueue	integer	
InfoType=AddressInfoAddressType			
AddressType			

Attribute InfoStatus	Attribute Extensions		
InfoType=AddressInfoCallsQuery			
NumberOfListElements	call-%d	integer	The Call ID of a call
	conn-%d	string	The Connection ID of a call (converted into a string)
	state-%d	integer	The state of the DN in question as a party of the call. See AddressStatusInfoType in the Voice Platform SDK API Reference for details.
InfoType=AddressInfoQueueLoginAudit			
NumberOfListElements	agent-extension	string	The agent ID
InfoType=AddressInfoNumberOfIdleClassifiers			
NumberOfIdleClassifiers	Idle	integer	NumberOfIdleClassifiers
	InUse	integer	NumberOfClassifiersInUse
InfoType=AddressInfoNumberOfClassifiersInUse			
NumberOfClassifiersInUse	Idle	integer	NumberOfIdleClassifiers
	InUse	integer	NumberOfClassifiersInUse
InfoType=AddressInfoNumberOfIdleTrunks			
NumberOfIdleTrunks	Idle	integer	NumberOfIdleTrunks
	InUse	integer	NumberOfTrunksInUse
InfoType=AddressInfoNumberOfTrunksInUse			
NumberOfTrunksInUse	Idle	integer	NumberOfIdleTrunks
	InUse	integer	NumberOfTrunksInUse
InfoType=AddressInfoDatabaseValue			
	ID	string	A database value

Example



EventAddressInfo Feature Example

EventPartyInfo

Description

The information about the call specified by `ConnID`. T-Server returns `EventPartyInfo` only to the client that issued a call to the `TQueryCall()` function.

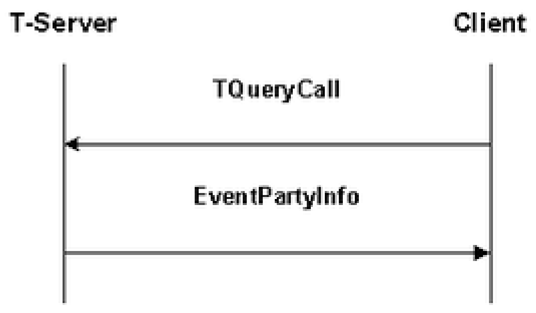
Contents

Event Attribute	Type
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions [a]	Mandatory
InfoStatus.NumberOfListElements [b]	Mandatory
NetworkCallID	Optional
NetworkNodeID	Optional
OtherDN [c]	Optional
OtherDNRole [c]	Optional
PreviousConnID [d]	Optional
ReferenceID	Mandatory

Event Attribute	Type
Server	Mandatory
ThisDN [e]	Optional
ThisDNRole [e]	Optional
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Optional

a. The directory numbers of parties participating in the call specified by `ConnID` are specified as key-value pairs with the key `party-<n>` in the `Extensions` attribute, where <n> is the party number. Some switches might not report a Queue or a Routing Point as a call party.
b. This attribute consists of the number of known parties participating in a call.
c. T-Server returns the `OtherDN` and `OtherDNRole` attributes if the condition described in Table Footnote a. takes place, and the number of parties involved in the call specified by `ConnID` is `2`.
d. The attribute must appear if the value of `CallType` is `Consult`.
e. T-Server returns the `OtherDN` and `OtherDNRole` attributes if the condition described in Table Footnote a. takes place, and the number of parties involved in the call specified by `ConnID` is `2`.

Example



EventPartyInfo Feature Example

EventLocationInfo

Description

This event is generated as a response to the `TQueryLocation()` function and includes information specified by `InfoType` about the remote location specified by the `Location` parameter.

Contents

Event Attribute	Type
Event	Mandatory
Extensions	Optional
InfoType	Mandatory
Location	Optional
Server	Mandatory
time	Mandatory
ReferenceID	Optional

Additional information based on `InfoType` is provided in the `Extensions` attributes. When information is requested about one remote location, the `Extensions` attribute can contain the following key-value pairs:

Extensions in EventLocationInfo

Key	Value	Value Type
LQ-location-name	location	string
LQ-location-status	loc-status, 0 - disconnected (configured) 1 - connected	integer
LQ-link-status	link-status, 0 - disconnected 1 - connected	integer

When information is requested about more than one location, these same key-value pairs for each location are referred to in the `loc-list[i]` value of the `LQ-location-%d` key within the `Extensions` attribute, where `i` is the number of a remote location (`i=0` defines the first location).

InfoType and Extensions in EventLocationInfo

Key	Value	Value Type

	InfoType=LocationInfoAllLocations or
	InfoType=LocationInfoMonitorAllLocations	

LQ-location-%d	loc-list[i]	kv-list

	InfoType=LocationInfoLocationData or
	InfoType=LocationInfoMonitorLocation	

LQ-location-name	location	string
LQ-location-status	loc-status, 0 - disconnected (configured) 1 - connected	integer

Key	Value	Value Type
LQ-link-status	link-status, 0 - disconnected 1 - connected	integer

EventServerInfo

Description

Delivers the information about T-Server specified in `ServerVersion`, `ServerRole`, and `Capabilities`.

Contents

Event Attribute	Type
Capabilities	Mandatory
CustomerID	Optional
Event	Mandatory
Extensions [a]	Mandatory
HomeLocation	Optional
ReferenceID	Mandatory
Server	Mandatory
ServerRole	Mandatory
ServerVersion	Mandatory
time	Mandatory
ServerCapabilityMask	Mandatory [b]

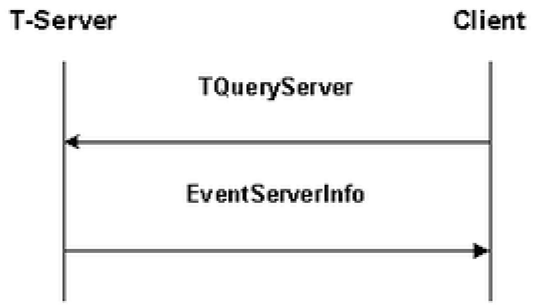
a. See Extensions in EventServerInfo for details. When related to information about a T-Server's ability to support transaction monitoring, this attribute indicates that support, and its key-value pairs contain information about the supported transaction monitoring classes. (For transaction monitoring purposes, this attribute is not present when the T-Server in question does not support that feature.)
b. If the capability mask includes `TransactionMonitoring` and `EventTransactionStatus`, the T-Server in question supports transaction monitoring.

Extensions in EventServerInfo

Key	Value	Value Type
T-Server	The full string designating information about the current version of T-Server (the same as if T-Server has been started with the `-V` command line option)	string

Key	Value	Value Type
Features	The list of features with which T-Server is built (the same list as that which results from T-Server being started with the `-V` command line option)	string

Example



EventServerInfo Feature Example

EventSwitchInfo

Description

This event is generated as a response to the `TQuerySwitch()` function and includes the requested information.

Contents

Event Attribute	Type
CustomerID	Optional
Event	Mandatory
Extensions	Mandatory
Reasons	Optional
ReferenceID	Mandatory
Server	Mandatory
time	Mandatory

Extensions in EventSwitchInfo

InfoStatus	Attribute Extensions		
Key	**Value Type**	**Value Description**	

InfoStatus	Attribute Extensions		
SwitchInfoDataTime	Date/Time	string	Current date and time information on the switch in the MM/DD/ YYYY HH:MM:SS format
SwitchInfoClassifierStat	Idle	integer	NumberOfIdleClassifiers
	InUse	integer	NumberOfClassifiersInUse

User-Data Events

EventAttachedDataChanged

Description

The attached data for the call has been changed.

Contents

Event Attribute	Type
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallType	Mandatory
ConnID	Mandatory
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
ReferenceID	Optional
Server	Mandatory
ThirdPartyDN [a]	Optional
ThisDN	Optional
ThisDNRole	Optional
ThisTrunk	Optional
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional
UserData	Mandatory

a. The `ThirdPartyDN` attribute is used to identify who initiated a change in user data. It is mandatory if it can be specified.

ISCC Events

EventAnswerAccessNumber

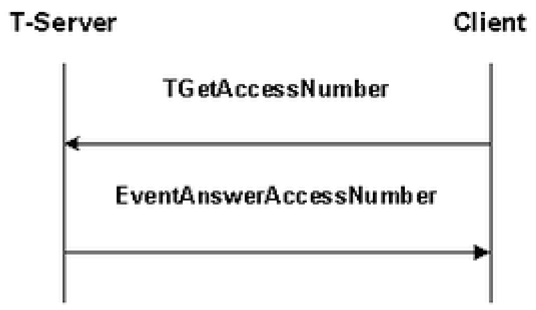
Description

T-Server has processed a previously called `TGetAccessNumber()` function and has returned the requested access number.

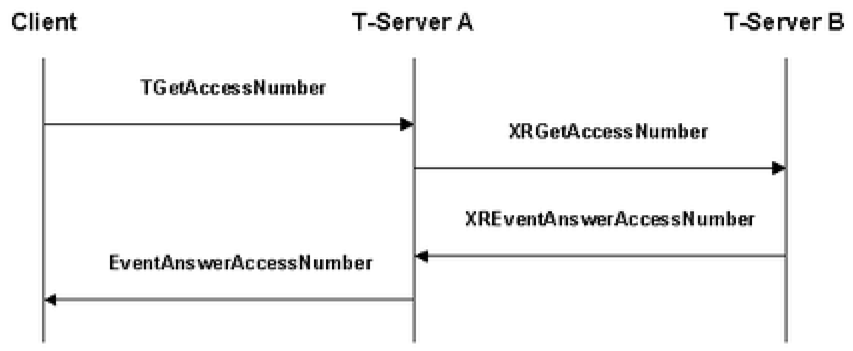
Contents

Event Attribute	Type
AccessNumber	Mandatory
ThisDN	Mandatory
ConnID	Mandatory
Event	Mandatory
ReferenceID	Mandatory
Server	Mandatory
time	Mandatory
Extensions	Optional

Examples



EventAnswerAccessNumber when GetAccessNumber Does Not Contain AttributeLocation



EventAnswerAccessNumber when TGetAccessNumber Contains AttributeLocation

Important
In `EventAnswerAccessNumber` T-Server A acts as an origination, and T-Server B a destination T-Server.

EventRemoteConnectionSuccess

Description

The call has successfully reached the remote switch.

Contents

Event Attribute	Type
ConnID	Mandatory
CustomerID	Optional
Event	Mandatory
Extensions	Optional
ReferenceID	Mandatory
Server	Mandatory
ThisDN	Mandatory
time	Mandatory

EventRemoteConnectionFailed

Description

The call failed to reach the remote switch or could not be treated as successful (for example, ISCC could not route the call to a requested destination DN).

Contents

Event Attribute	Type
ConnID	Mandatory
CustomerID	Optional
Event	Mandatory
ReferenceID	Mandatory
Server	Mandatory
time	Mandatory
ThisDN	Mandatory
Extensions	Optional

EventReqGetAccessNumberCanceled

Description

T-Server has canceled a previously called `TGetAccessNumber()` function before processing is complete.

Contents

Event Attribute	Type
Event	Mandatory
ReferenceID	Mandatory
Server	Mandatory
time	Mandatory
XReferenceID	Mandatory
Extensions	Optional

Special Events

EventACK

Description

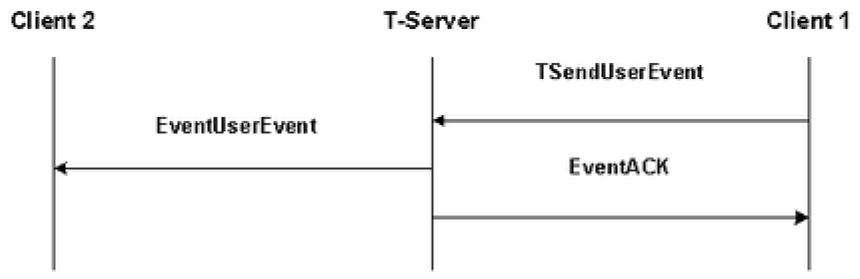
T-Server has acknowledged a request received from a client application. This event is a response to the `TSendUserEvent()`, `TSendEvent()`, `TSendEventEx()`, `TSetInputMask()`, `TTransactionMonitoring()`, and `TPrivateSevice()` functions.

Contents

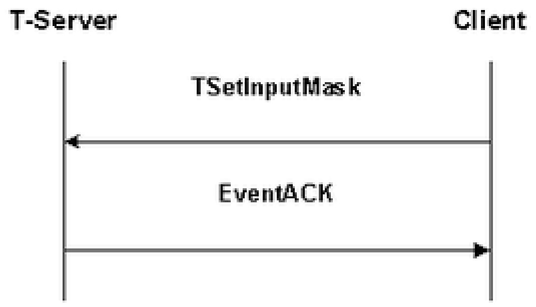
Event Attribute	Type
Event	Mandatory
ReferenceID	Mandatory
Server	Mandatory
time	Mandatory
UserEvent [a]	Optional
Extensions	Optional
SubscriptionID	Optional [b]

a. `UserEvent` is the identifier of the request that has caused T-Server to generate `EventACK`. The attribute is derived from the request.
b. This attribute, applicable only to `EventACK` in response to `TTransactionMonitoring()`, is mandatory for the operation `SubscriptionStart`, but is not present for the operations `SubscriptionStop` and `SubscriptionModify`. This attribute represents a subscription identifier: the call has successfully reached the remote switch.

Examples



EventACK Feature, Example 1



EventACK Feature, Example 2

EventAgentReserved

Description

This event is generated as a positive response to the `TReserveAgent()` request, confirming that those of the parameters `agent_dn`, `agent_id`, and `agent_place` that were present in the request have been successfully reserved. When an agent becomes reserved as the result of `TReserveAgent()` request, the event is sent to the application that sent the request and to all clients registered on the agent's DN.

Contents

Event Attribute	Type
AgentID [a]	Optional
Event	Mandatory
Place [a]	Optional
Reasons	Optional
ReferenceID	Optional
Server	Mandatory
ThisDN [a]	Optional
Timeout [b]	Optional
time	Mandatory
Extensions	Optional

a. At least one of the `AgentID, Place,` or `ThisDN` attributes must be present. The attributes are equal to the corresponding parameters of the `TReserveAgent()` function.
b. The duration of the reservation is in milliseconds.

EventCallInfoChanged

Description

The call information has been changed.

Contents

Event Attribute	Type
ConnID	Mandatory
CustomerID	Optional
Event	Mandatory
Extensions	Optional
ReferenceID	Optional
Server	Mandatory
time	Mandatory

EventPrivateInfo

Description

This event is generated both in response to a `TPrivateService()` function call and when it is used to notify selected sets of clients of T-Server–specific information. This event indicates that one of two forms of data has been passed:

- Information supported only by certain T-Servers, and which is not covered by general feature requests.
- Notification about changes in T-Server behavior or device/call statuses.

This event can be distributed to the following clients:

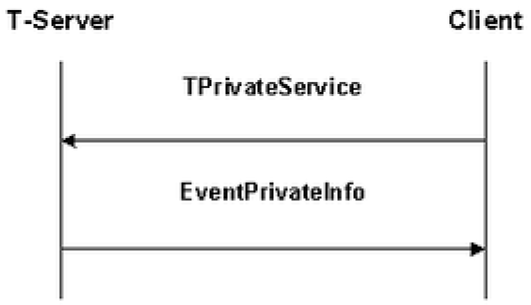
- Those who made a request for the private service, `TPrivateService()`.
- Those registered on the specified device, depending on the nature of the service.
- Those who would otherwise be ineligible (for security reasons) for receiving given events, and who thus require additional registration in order to be notified.

Contents

Event Attribute	Type
PrivateEvent	Mandatory
Event	Mandatory

Event Attribute	Type
Reasons	Optional
UserData	Optional
Extensions	Optional
ReferenceID	Optional
Server	Mandatory
ConnID	Optional
ThisDN	Optional
time	Mandatory

Example



EventPrivateInfo Feature Example

EventUserEvent

Description

A user event from another client application has been received.

Contents

`EventUserEvent` contains T-Server-required attributes. Other attributes are specified by the initiator of this event.

Event Attribute	Type
CustomerID	Optional
Event	Mandatory
Server	Mandatory
ThisDN	Mandatory
time	Mandatory

Event Attribute	Type
Extensions	Optional

EventPrimaryChanged

Description

A change in the role of one of a pair of synchronized T-Servers in hot standby mode has taken place: the T-Server's role has been changed from backup to primary.

Warning

After a client receives `EventPrimaryChanged`, it should re-synchronize with the new primary T-Server (the one reporting this event) by calling the `TQueryLocation()` function and either the `TQueryAddress()` or `TQueryCall()` function.

If T-Server issues `EventPrimaryChanged`, and the two T-Servers have different link statuses at the time of switchover, `EventLinkDisconnected` or `EventLinkConnected` is generated as follows:

- If the former primary T-Server has its link connected, and the new one does not, `EventLinkDisconnected` is delivered just prior to `EventPrimaryChanged`.
- If the former primary T-Server has its link disconnected, and the new one has its link connected, `EventLinkConnected` is delivered immediately after `EventPrimaryChanged`.

The ordering of these messages is intended to simplify the processing logic on the client side—receiving `EventPrimaryChanged` when the link is disconnected may be safely ignored.

Important

The event generated for a primary-to-backup switchover—`EventRestoreConnection`— is different from `EventPrimaryChanged` in that it is processed internally by T-Library and is never delivered to clients.

Contents

Event Attribute	Type
ApplicationName	Optional
time	Mandatory

EventRestoreConnection

Description

A synchronized connection between a pair of T-Servers in hot standby mode has been established, or the T-Server's role has been changed from primary to backup.

Contents

Important

`EventRestoreConnection` is processed internally by T-Library, and is not delivered to the user dispatch function. As such, it does not use the conventional event contents structure.

Attributes

- `ServerRole`—The primary T-Server generates `EventRestoreConnection`, with the attribute `ServerRole` set to `0`,when the connection to the backup T-Server has been established.
  T-Server generates `EventRestoreConnection` and sets the attribute `ServerRole` to `1` to indicate that it has been changed from the primary to the backup T-Server. This value is used to prevent a race condition in situations where two or more switchovers happen within a short interval of time. (Otherwise, T-Library relies on `EventPrimaryChanged` to identify the primary T-Server.)
- `UserData`—This attribute contains the reference to the backup T-Server. T-Library uses this information to try to connect to that server.

EventHardwareError

Description

T-Server has detected inconsistent data or an incomplete event flow in switch messaging—for instance, a wrong checksum or missing attributes.

Contents

Event Attribute	Type
CustomerID	Optional
ThisDN	Optional
ConnID	Optional
PreviousConnID	Optional

Event Attribute	Type
ErrorCode	Optional
Event	Mandatory
Server	Mandatory
time	Mandatory
Extensions	Optional

EventResourceAllocated (Obsolete—No Longer Supported)

Description

The switch has registered the CTI link to receive events about the specified telephony object.

EventResourceFreed (Obsolete—No Longer Supported)

Description

The switch has canceled registration of the CTI link, so that it no longer can receive events about the specified telephony object.

Negative-Response Events

EventError

Description

The requested function cannot be performed. The reasons are specified by the values of the `ErrorCode` and `ErrorMessage` parameters.

Contents

Event Attribute	Type
ConnID	Optional
CustomerID	Optional
ErrorCode	Mandatory [a]
ErrorMessage	Optional
Event	Mandatory
ReferenceID	Mandatory
Server	Mandatory
ThisDN	Optional
time	Mandatory
Extensions	Optional [b]

a. `ErrorCode`, when received as a negative response to `TTransactionMonitoring()`, may include the following reasons:

- `TERR_UNSUP_OPER` (unsupported operation), the Transaction Monitoring Feature is not supported by this T-Server.
- `TERR_INVALID_ATTR` (invalid attribute value), the transaction monitoring request contains an invalid attribute.

b. Contains additional information on a failure.

Important

The `Reasons` attribute was formerly included in this event. It is now obsolete. (It is omitted from the `EventError` message since, presumably, the requestor is aware of the reason for the request.)

Agent States and Work Modes

Agent states specify what state an agent is in. For example, an agent in Ready state is available to handle calls from an ACD queue. An agent can have several states with respect to different ACD devices, or he can use a single state to describe his relationship to all ACD devices. Agent states are reported in agent-state events.

Agent-State Diagram shows the agent states. Transitions between states, represented by arrows, show subsequent states that may be entered from a given state. The agent-state change occurs after T-Server generates a proper event.

Agent Null

The state where an agent is not logged in to an ACD group. Logging on and logging off cause the transition to and from this state.

Agent Not Ready

The state where an agent is logged in to an ACD group, but is not prepared to handle calls that the ACD distributes. While in this state, an agent can receive calls that are not handled by the ACD.

Agent Ready

The state where an agent is logged in to an ACD group and is prepared to handle calls that the ACD distributes.

Agent Busy Ready

The state where a device, on behalf of an agent, is involved with an existing ACD call even if that call is on hold at this device. After the call has been completed, the device is placed in the `Agent Ready` state (that is, it can pick up a new call). Direct calls between agents, calls between supervisors and agents, and private calls do not cause this transition.

Agent Busy NotReady

The state where a device, on behalf of an agent, is involved with an existing ACD call even if that call is on hold at this device. After the call has been completed, the device is placed in the `Agent Not Ready` state (that is, it cannot receive further calls from the ACD). Direct calls between agents, calls between supervisors and agents, and private calls do not cause this transition.

Agent Busy AfterCallWork

The state where a device, on behalf of an agent, is involved with an existing ACD call even if that call is on hold at this device. After the call has been completed, the device is placed in the `Agent After Call Work` state (that is, it is not able to receive further calls). Direct calls between agents, calls between supervisors and agents, and private calls do not cause this transition.

Agent After Call Work

The state where a device, on behalf of an agent, is no longer involved with an ACD call. While in this state, the agent is performing administrative duties for a previous call and cannot receive further calls from the ACD.

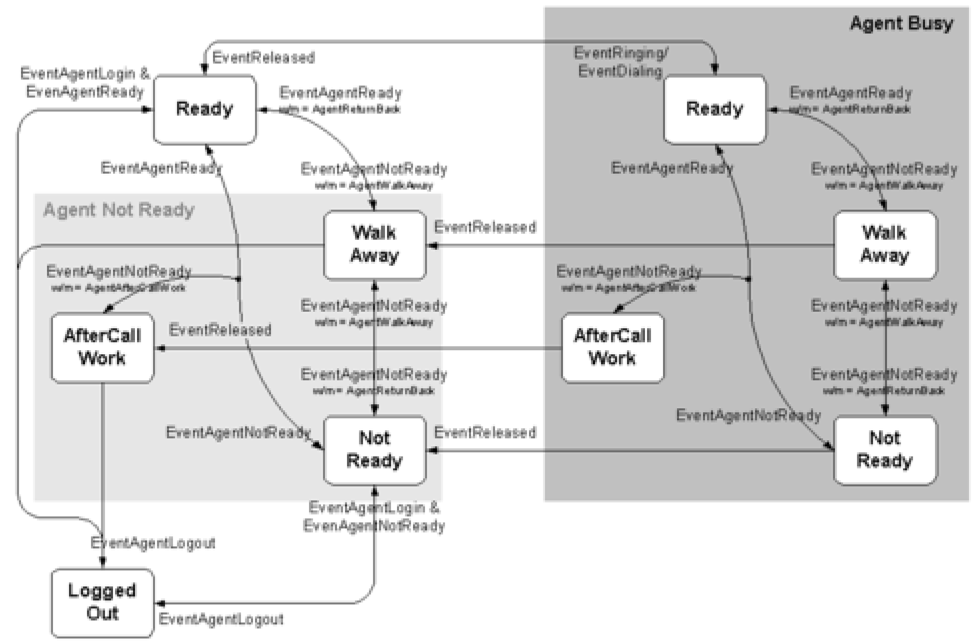
Agent Return Back

The state where an agent is logged in to an ACD group upon having returned from the `WalkAway` state. The agent may or may not be ready to receive calls.

Agent Walk Away

The state where an agent is logged in to an ACD group, but is understood not to be at his station, and thus not prepared to handle calls that the ACD distributes.

Agent-State Diagram


Agent Busy
EventReleased
EventRinging/ EventDialing
EventAgentLogin & EvenAgentReady
Ready
EventAgentReady w/m = AgentReturnBack
EventAgentReady
EventAgentNotReady w/m = AgentWalkAway
Agent Not Ready
Walk Away
EventReleased
EventAgentNotReady w/m = AgentAfterCallWork
EventAgentNotReady w/m = AgentWalkAway
AfterCall Work
EventReleased
EventAgentNotReady w/m = AgentReturnBack
EventAgentNotReady
Not Ready
EventReleased
EventAgentLogin & EvenAgentNotReady
EventAgentLogout
Logged Out
EventAgentLogout
Ready
EventAgentReady w/m = AgentReturnBack
EventAgentReady
EventAgentNotReady w/m = AgentWalkAway
Walk Away
EventAgentNotReady w/m = AgentAfterCallWork
AfterCall Work
EventAgentNotReady w/m = AgentWalkAway
EventAgentNotReady w/m = AgentReturnBack
EventAgentNotReady
Not Ready

Agent-State Diagram

Agent-State Diagram Comments

In certain cases, even though the agent state remains the same, T-Server might generate distinct events to represent internal state transitions or data changes within this state. This can take place if T-Server is required to support switch-specific substates or if it must distribute additional resources or extensions. The following rules apply to these existing-state transitions:

- T-Server filters out unsolicited CTI messages that do not provide new information from the last

`EventReady/NotReady`. For instance, if the switch re-sends the exact same event every time an ACD call is released on an agent's DN, T-Server does not continue to pass on these events to its clients. On the other hand, if switch-specific sub-states, as reported in `AttributeWorkMode`, or if the `ReasonCode` attribute is changed, then T-Server distributes `EventReady/NotReady`.

- T-Server distributes a corresponding `EventReady/NotReady` when clients initiate a same-state transition, provided T-Server can confirm the state with the switch, or T-Server has enough confidence that the internally kept state is correct. In the process of confirming the state with the switch, if T-Server receives an error message that suggests "in the same state," it translates this into a positive response. If these CTI messages do not provide adequate information regarding the problem with the request, based on internal data, T-Server may generate a response to the client on its own.

> **Important**
>
> Beginning with the 7.1 release of T-Server, the function TAgentSetIdleReason(), and its corresponding event, is no longer supported. Instead, use the generic Request/EventAgentReady/NotReady with a new value for ReasonCode in the Extensions attribute and/or in the attribute Reason.

- The following transitions are available as of the 7.1 release of T-Library:
  - T-Server responds with `EventAgentLogin` (distributed to all registered clients) in response to the `AgentLogin()` function, as long as the credentials (`AgentID` and `ThisQueue`) used by the already-logged-in agent are the same for this DN (and provided T-Server is able to verify this information on the switch). In the case of credentials that differ:
    - Since the Genesys model supports only one agent with distinct `AgentID` logged in on a given device, depending on the switch, T-Server either rejects a request with a distinct `AgentID` or forces a logout of the old `AgentID`.
    - Requests that have different `ThisQueue` values are processed according to whether multiple logins are allowed by the CTI protocol.
- `EventAgentLogout` in connection with a `Logged Out` state is allowed in response to a client's request, but is not required. Genesys recommends that you direct the request to the switch and translate any positive acknowledgement (one indicating a previous agent state de-synchronization) into `EventAgentLogout`.

Agent-State Diagram Limitations

- Not all agent states are available on all switching platforms. Furthermore, some platforms may have additional substates. Please refer to the switch's documentation for more information.
- Depending on switch capabilities, not all transitions are available on all platforms. Please refer to the switch's documentation for more information.
- In some cases, when T-Server does not have sufficient information from the CTI link, it may use the `Logged In` state (not shown in the diagram) to indicate that an agent is logged in, but that his status is not known. In order to report the transition from the `Logged Out` to the `Logged In` state, T-Server sends a single `EventAgentLogin`,without also sending an `EventReady` or `EventNotReady`.

Unified Call-Party States

A *call-party* is the relationship between a call and a given telephony device. Call-Party is often referred to as party. For the purposes of this section, the two terms are interchangeable.

A *call* is typically an interaction between two or more telephony objects (or between a telephony object and a network entity) that is established by the use of telephony network capabilities. However, in fact, Genesys assumes this association may have zero to many parties. In the context of the Genesys model, a call is a stateless object, and it always has a unique connection ID.

A *party* (call-party) represents the call's relationship to a given telephony device—either an internal DN or an external (out-of-PBX) call participant. Each call-party has state (and a number of other attributes).

A *call-party state* is a packaging of the fuller expression of a party's status (which may be multifaceted)—a compound state made up of the following groups:

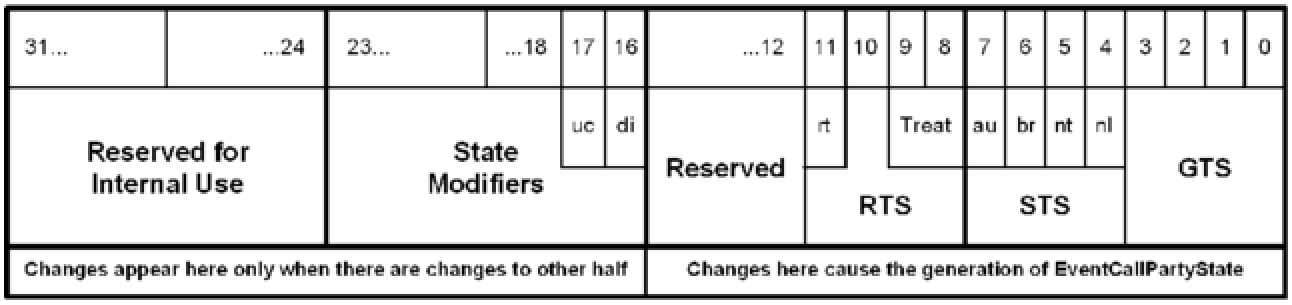
- Generic Telephony State (GTS);
- Supplementary Telephony State (STS);
- Routing/Treatment State (RTS).
- State modifiers

Each of these groups has parts that are referred to as elementary states, or properties (the basic attribute of a call-party). Typically, a call-party state consists of just one of these groups. However, there are certain cases where a coexistence of multiple groups in one call-party state is the norm. These, for instance, should be familiar:

- Routing with Queueing involves GTS and RTS.
- Service Observing involves both GTS and STS.

> **Important**
> Although not all combinations of elementary states make sense, there is no restriction on them.

A state is packed into one integer according to the structure shown below:


31...
...24
23...
...18
17
16
...12
11
10
9
8
7
6
5
4
3
2
1
0
Reserved for Internal Use
State Modifiers
uc
di
Reserved
rt
Treat
RTS
au
br
nt
nl
STS
GTS
Changes appear here only when there are changes to other half
Changes here cause the generation of EventCallPartyState

Call-Party Structure

Availability of State Information

The Genesys Framework assumes a particular call-party state for each call-party in the enterprise and supports the generic party-state set common to those T-Servers that support it. Not all party states or party-state transitions are mandatory for each T-Server. A given vendor-specific CTI may not provide a specific call-party state. Or, if a given state is available with the CTI, it may lack some relevant information required for T-Server to represent that particular state on the Genesys side of the software, or permit Genesys to perform a party-state transition. Thus the implementation of call-party states in T-Server is only complete to the extent that information is available from the switch. See `PartyState` in your API reference for details and state numeric values.

Generic Telephony State

The *Generic Telephony State* part of a call-party state comprises the most important of the sub-party states.

Important

Previously, the Genesys model did not include the states `Initiated` and `Failed` as marking participation in a call. As a result, there were no special events in DN-based event reporting to indicate a transition between those states and `NULL`. Some T-Servers used device-related `EventOffHook/OnHook` with at tribute `ConnID` to indicate a party had been added or deleted, but this event is optional and cannot be used when there is another active call on the same device (otherwise the status of the device is reported incorrectly). It is not possible to reconstruct the `Initiated` and `Failed` states from DN-based reporting. Furthermore, for compatibility reasons, parties in these states are not included in the `EventPartyInfo` list, and related calls are not included in `EventRegistered` and `EventAddressInfo`.

The Generic Telephony State has the following possible properties:

Null and Null^2—Represent intermediate states when party information exists in T-Server memory, but when there is no logical relation between a call and device.

Initiated—Indicates that a call has been initiated by a device. Any telephony device that is initiating a new call enters the `Initiated` state while it awaits the availability of switch resources or user input. That is, the device remains in the `Initiated` state from the moment it goes off-hook until `EventDialing` is sent.

Queued—Identifies that a call is queued or parked on a given device, and that it awaits the availability of some service (for example, ACD queue distribution) or of some device (for example, a phone line).

Alerting—Means that a call is alerting on a device, indicating an incoming call (for example, the phone is ringing), or that a call is in the process of being distributed to a destination (for example, is being processed by the telephony network).

Connected—Indicates that a given device has a voice connection with other participants.

Important

The `Dialing` state (which is implicitly used in the DN call model) does not exist in the unified party-state model. Dialing was an attempt to report the state of the other party on the call. Such other parties cease to be clear in complex scenarios where transfers and conferences confuse matters. However, `Dialing`, as a state modifier, is available for compatibility with traditional reporting.

Call Progress (CP) Detection—The originating device (either a queue or a route point) for predictive dialing is put in this state from the time a call is made (and `EventDialing` is sent) to the moment the call is classified, at which time it is either moved to the `Queued` state, or is released.

Held—A device has temporarily suspended its connection to other call participants.

Busy—A call cannot reach the intended device, which is busy.

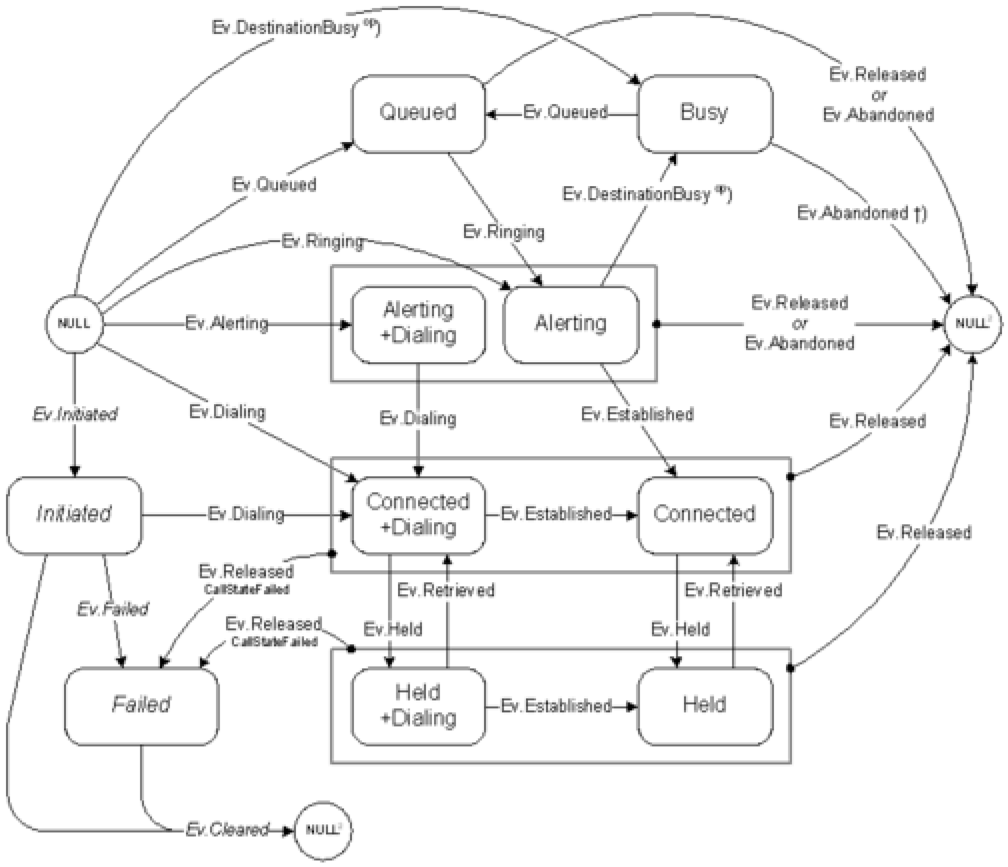
Failed—Indicates that a call originating on a given device has not succeeded (either the dialed number is wrong, or the switch was not able to allocate the trunk). This state is used instead of `Busy` when a destination party was never created for the call. `Failed` also applies to a party whose call has been disconnected, but who remains off hook.

Generic Telephony State Diagram for Regular DNs

The Generic Telephony State for Regular DNs diagram indicates the event flow that leads to the various sub states that comprise the Generic Telephony State. This diagram applies to all regular DNs. That is, all types except ACD queues and route points.

Important

Although they appear in the Generic Telephony State for Regular DNs diagram, events `Alerting`, `Initiated`, `Failed`, and `Cleared` do not currently exist. These names are reserved for future use.



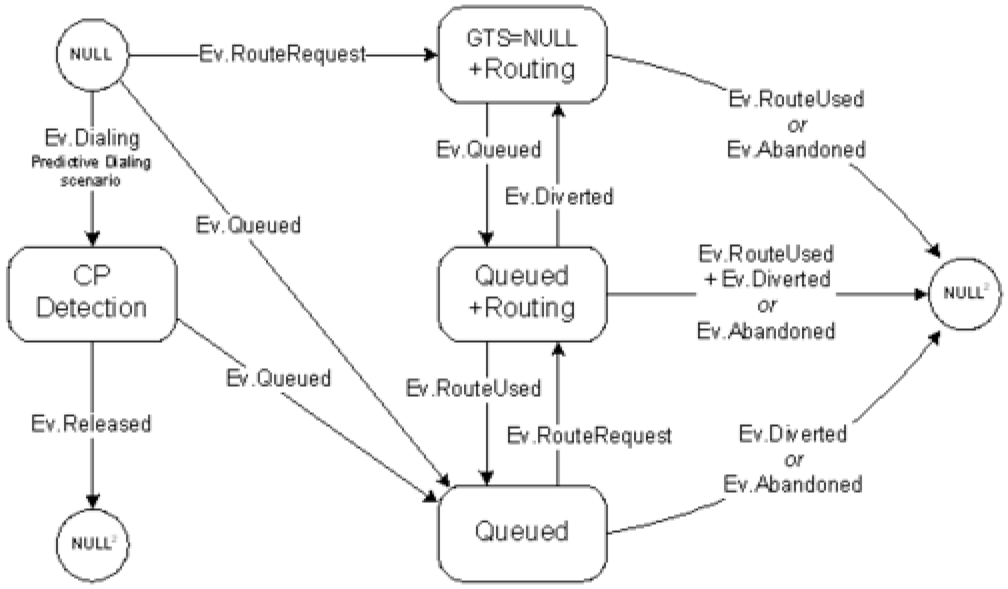
Generic Telephony State for Regular DNs

op Note that in this case, with a DN-based call model, `EventDestinationBusy` is reported for the opposite party.

† `EventAbandoned` is only sent if there is an `EventRinging` for that party. This occurs only with external parties.

Generic Telephony State Diagram for ACD Queues and Route Points

The Generic Telephony State for ACD Queues and Route Points diagram indicates the event flow that leads to the various sub states that comprise the Generic Telephony State. This diagram applies to ACD queues and route points.



Generic Telephony State for ACD Queues and Route Points

Supplementary State

The *Supplementary State* refers to combinations of the following party properties. Not all combinations of these properties make logical sense, but the general model does not put any restrictions on them. Furthermore, there is no transitional model for these properties (any state can change to any other).

NoListen—A party cannot hear other participants on the call.

NoTalk—A party is muted, and other participants on the call do not hear that party.

Audit—A party is attached to a call as `Service Observer` or `Service Assistant.`

Bridged—A party is attached to the call with the Bridged Call Appearance feature.

State Modifiers

State Modifiers further nuance the relationships between events and call states by allowing for intermediate sub states. These sub states provide additional information to the system, but may otherwise be unnecessary in the unified call-party state model.

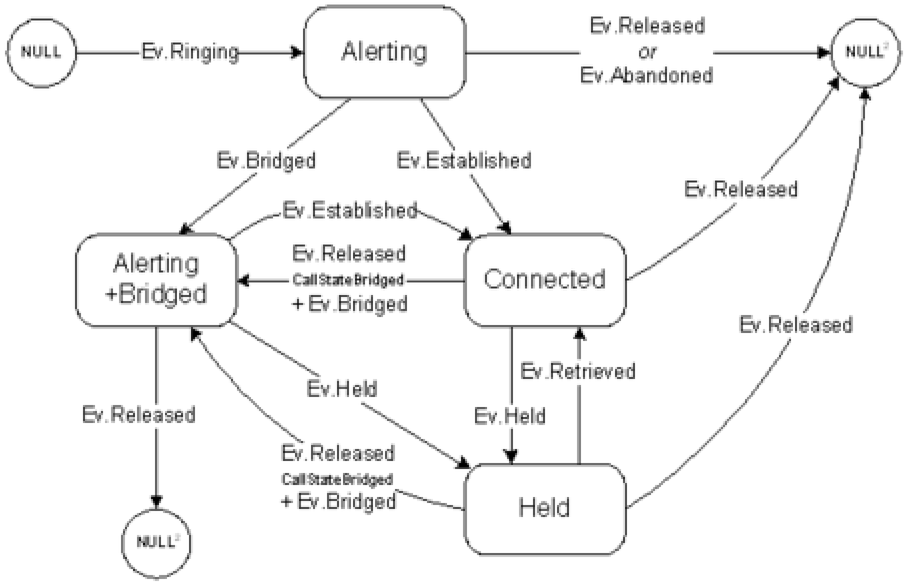
Dialing Modifier

The unified call-party state model uses `Dialing`, although not actually a party state, as a state modifier. This allows the system to handle the following scenarios, also illustrated in Modifier Dialing:

- Reporting applications may need to measure dialing time, which is defined as the interval between the moment a party is connected to a call and the moment when the voice connection with the other participant is established for the first time. It is possible to reconstruct the time spent in a dialing state without this modifier, but, since T-Server does not provide historical data, this calculation needs to be done when the full history of the call is available (by factoring in the states of other parties). The `Dialing` modifier provides clients access to this information during the call.
- T-Server needs an indication of whether `EventEstablished` has been distributed for a given party. In a DN-based call model this event is sent only when a connection is first established. (See Modifier Dialing for an instance of why a subsequent EventEstablished would be useful.) While it is possible to have this indication stored in device-specific states, the `Dialing` modifier allows for common functionality across media devices.

Uncertain Modifier

This modifier is set when the party information is not reliable. See `Reliability` in your API reference for details.



Modifier Dialing

Routing/Treatment State

The *Routing/Treatment State* consists of the following properties:

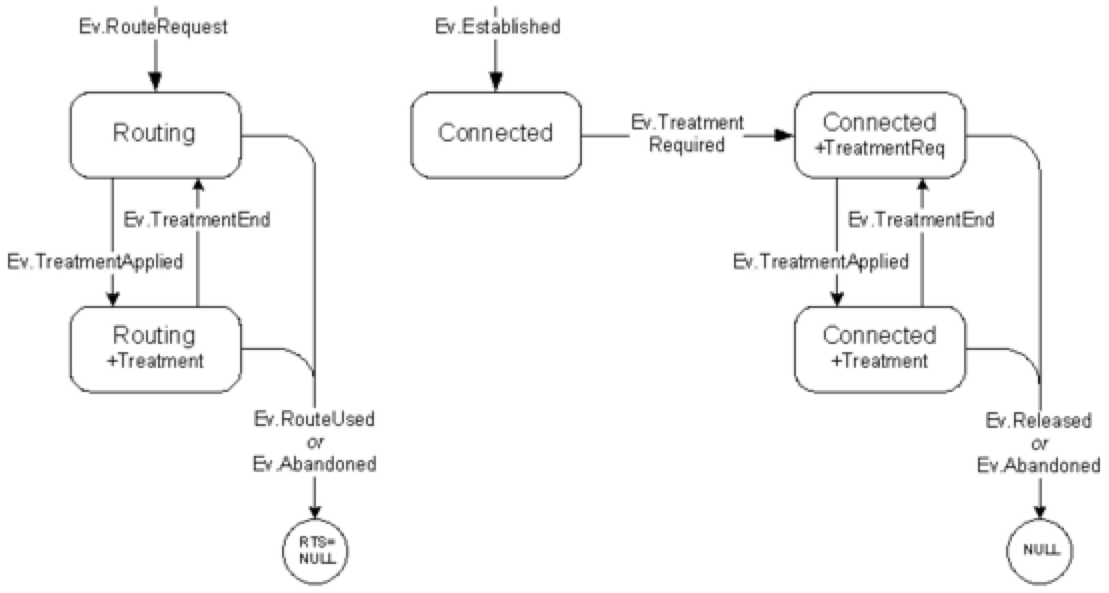
TreatmentReq—The switch is waiting for a treatment to be applied to the call.

Treatment—A treatment has been applied while the call is located on a given device.

Routing—The switch is waiting for routing instructions.

Routing/Treatment State Diagram

The following figure indicates the event flows that lead to the various states comprising the `Routing` (left portion of the diagram) and `Treatment` (right portion of the diagram) states. The `Routing` portion pertains to route points; the `Treatment` portion pertains directly to monitored IVR ports.



Routing (left portion) and Treatment States (right portion)

Event Attributes

This section describes the attributes that make up each event.

AccessNumber

A pointer to a number by dialing which a client application from the specified switch can reach a specific external routing point (`ThisDN`).

AgentID

This parameter uniquely identifies the ACD agent. For more information, refer to the type `AgentID` in your API reference.

ANI

Automatic Number Identification. Indicates the telephony-company charge number.

CallHistory

Information about transferring/routing of the call through a multi-site contact center network. For more information, refer to the type `CallHistoryInfo` in your API reference. Typically used to keep track of a call in multi-site contact centers.

CallID

This attribute contains the call identification provided by the switch, which uniquely identifies a call. As opposed to `ConnID` assigned by T-Server, `CallID` is created by the switch when the incoming call arrives, or when agent/system out-dial calls are created. The attribute must be present if the switch generates and distributes the corresponding parameter to T-Server. (`CallID` is zero as long as the switch does not provide that information to T-Server.) For more information, refer to the type `CallID` in your API reference.

CallingLineName (Obsolete)

A pointer to the name of the person associated with the directory number from where the inbound call in question has been made. It can be distributed by an originating switch through an ISDN trunk only.

CallState

The current status of the call the event relates to. For more information, refer to the type `CallState` in your API reference.

CallType

The type of the call in question. For more information, refer to the type `CallType` in your API reference.

Capabilities

Switch-specific mask specifying the set of requests and events that this T-Server can handle.

Cause

For network calls, the reason for transitions to certain states—`Routing` and `NoParty.` (This helps clarify delivery failure, such as `Busy` or `NoAnswer.`)

CLID (Obsolete)

Calling Line Identifier. The directory number from where the inbound call was made.

CollectedDigits

A pointer to the digits that have been collected from the calling party.

ConnID

A current connection identifier of the call to which this event relates.

Connection ID Structure

Byte	Bits							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	Reserved		Global Server Identifier					
1	Global Server Identifier							
2	Local Connection Identifier							
3	Local Connection Identifier							
4	Local Connection Identifier							
5	Local Connection Identifier							
6	Local Connection Identifier							

Byte	Bits
7	Local Connection Identifier

ConnID Parameters

Reserved (bits `0` and `1`)—Bits reserved for future usage.

Global Server Identifier (bits `2-15`)—Unique identifier for this T-Server specified by the `server-id` option. If `server-id` is not specified, the `ConnID` may not be unique within a multi-site contact center.

Local Connection Identifier (bits `16-63`)—Local identifier of the call this event relates to.

For more information, refer to the type `ConnectionID` in your API reference.

CustomerID

A pointer to the string containing the assigned Customer (`Tenant`) identifier through which the processing of the call was initiated. The attribute must be present in every event for a multi-tenant contact center.

DNIS

Directory Number Information Service. The directory number to where the inbound call has been made.

ErrorCode

This attribute contains a value that indicates why a client request failed.

ErrorMessage

A pointer to the character string containing additional information about an error.

Event

The event identifier. For more information, refer to the type `MessageType` in your API reference.

Extensions

A pointer to an additional data structure that takes into account switch-specific features that cannot be described by the other parameters in an event or a request. Extensions that are specific to particular events are noted with their event information in the T-Library Events section. Some extensions for requests, however, are applicable to all T-Servers, and permit tuning of T-Server operations.

FileHandle

The handle of the voice file in question. For more information, refer to the type `File` in your API reference.

HomeLocation

A pointer to the name of the host where T-Server is running.

InfoStatus

The `InfoType` information about the telephony object specified by `ThisDN` and/or `ThisQueue`.

InfoType

The type of information about the telephony object in question.

LastCollectedDigit

The last digit collected from the calling party.

Location

The remote location's name, in the form of
`<Switch Name>`
or
`<T-Server Application Name>@<Switch Name>`
(Switch Name and T-Server Application Name are as defined in the Configuration Layer.)

LocationInfoType

A type of information requested by a client in the `TQueryLocation()` request. For more information, refer to the type `LocationInfoType` in your API reference.

NetworkCallID

In case of network routing, the call identifier assigned by the switch where the call initially arrived.

NetworkCallState

The current status of the network call the event relates to. For more information, refer to the type `NetworkCallState` in your API reference.

NetworkDestDN

The intended destination of the network operation. This may be in the form: `location::DN`.

NetworkDestState

The state of the destination party for a network call.

NetworkNodeID

In case of network routing, the identifier of the switch where the call initially arrived.

NetworkOrigDN

DN of the internal origination party in the form `location::DN`. This attribute is only available for first-party operations, those made on behalf of Agent 1, requested through a premise T-Server.

NetworkPartyRole

The role of a call participant with respect to an in-progress network-transfer request. For the agent who originated the last network-transfer request, the value is `RoleNtwkOrigParty`. For the new destination, the value is `RoleNtwkDestParty`. (Network T-Servers do not send this attribute.)

NodeID

Uniquely identifies a switch within a network.

OtherDN

The directory number of the second most significant telephony object (except an ACD group or trunk group) with respect to the event in question. The application does not have to be registered to this directory number to receive the event in question.

OtherDNRole

The role of the telephony object specified by `OtherDN` in the event in question. For more information, refer to the type `DNRole` in your API reference.

OtherQueue

The directory number of the second most significant ACD group with respect to the event in question.

OtherTrunk

The identifier of the second most significant trunk group with respect to the event in question.

Place

The identifier of the place requested for reservation.

PreviousConnID

This attribute links two associated calls. For example, events related to an original call include the connection ID of a consultation call; events related to a consultation call include the connection ID of the original call. See ConnID.

Warning

When `EventPartyChanged` is generated for the party that is still only involved in an original call (that is, `ConnID` has not been changed during a two-step operation), the `PreviousConnID` attribute is equal to `ConnID` of the original call.

PrivateEvent

The private event identifier. For more information, refer to the type `PrivateMsgType` in your API reference.

Reasons

A pointer to an additional data structure that provides reasons for and results of actions taken by the user of `ThisDN`. Any `Reasons` attribute that appears in `TEvent` is taken directly from the corresponding request. (See ReferenceID in Events That Correspond to Requests.) There is no other source for the information found in the content of the `Reasons` attribute. That is, no `Reasons` attribute should be expected for an event that is unsolicited. An event with no reference ID has no identifiable request that prompted it. See Persistent Reasons for more information.

(Switch information of a similar nature to this Genesys `Reasons` attribute is sometimes available, but those switch reasons are passed in the `Extensions` attribute.)

RefConnID

This attribute identifies the connection ID that results from the merging of two calls.

ReferenceID

`ReferenceID` is the identifier generated by T-Library or a `TSetReferenceID()` function call and attached to the request a client sends to T-Server. Every time a client sends a request to T-Server, it uses the current `ReferenceID` (increasing it by one each time). In response, T-Server generates an event. Only in the response to the client who initiated the request, as acknowledgment that the request has been fulfilled, the resulting event includes the same `ReferenceID` that was attached to the request. If the request fails, `EventError` will be sent only to the requestor. The ReferenceID in Events That Correspond to Requests table lists the events in which you will find the `ReferenceID` corresponding to that found with the request that prompted its assignment initially.

Important

For a limited number of specific requests, as noted in the ReferenceID in Events That Correspond to Requests table, T-Server may send more than one event with the same `ReferenceID`.

ReferenceID in Events That Correspond to Requests

Request	Event
General Requests	
TOpenServer	Not Applicable
TOpenServerEx	Not Applicable
TDispatch	Not Applicable
TCloseServer	Not Applicable
TScanServer	Not Applicable
TScanServerEx	Not Applicable
TSetInputMask	EventACK
Registration Requests	
TRegisterAddress [a]	EventRegistered
TUnregisterAddress [a]	EventUnregistered
Call-Handling Requests	
TAnswerCall	EventEstablished
TClearCall	EventReleased
THoldCall	EventHeld
TMakeCall [b]	EventDialing
TMakePredictiveCall	EventDialing
TReleaseCall	EventReleased
TRetrieveCall	EventRetrieved
TRedirectCall	EventReleased
Transfer/Conference Requests	
TInitiateConference [b]	EventDialing
TInitiateTransfer [b]	EventDialing
TCompleteConference	EventReleased
TCompleteTransfer	First arriving EventReleased
TDeleteFromConference	EventPartyDeleted or EventReleased

Request	Event
TReconnectCall	EventRetrieved
TMergeCalls	EventReleased
TMuteTransfer [b]	EventDialing
TAlternateCall	EventHeld
TSingleStepConference	EventPartyAdded or EventRinging
TSingleStepTransferb	EventReleased
Network Transfer/Conference Requests	
TNetworkConsult	EventNetworkCallStatus
TNetworkAlternate	EventNetworkCallStatus
TNetworkTransfer	EventNetworkCallStatus
TNetworkMerge	EventNetworkCallStatus
TNetworkReconnect	EventNetworkCallStatus
TNetworkSingleStepTransfer	EventNetworkCallStatus
TNetworkPrivateService	EventNetworkPrivateInfo
Call-Routing Requests	
TRouteCall [b]	EventRouteUsed
Call-Treatment Requests	
TApplyTreatment	EventTreatmentApplied+ EventTreatmentEnd or EventTreatmentNotApplied
TGiveMusicTreatment	EventTreatmentApplied
TGiveRingBackTreatment	EventTreatmentApplied
TGiveSilenceTreatment	EventTreatmentApplied
DTMF Requests	
TCollectDigits	EventDigitsCollected
TSendDTMF	EventDTMFSent
Voice-Mail Requests	
TOpenVoiceFile	EventVoiceFileOpened
TCloseVoiceFile	EventVoiceFileClosed
TLoginMailBox	EventMailBoxLogin
TLogoutMailBox	EventMailBoxLogout
TPlayVoice	EventVoiceFileEndPlay

Request	Event
Agent and DN Feature Requests	
TAgentLogin	EventAgentLogin
TAgentLogout	EventAgentLogout or EventQueueLogout
TAgentSetReady	EventAgentReady
TAgentSetNotReady	EventAgentNotReady
TCallSetForward	EventForwardSet
TCallCancelForward	EventForwardCancel
TMonitorNextCall	EventMonitoringNextCall
TCancelMonitoring	EventMonitoringCancelled
TSetMuteOff	EventMuteOff
TSetMuteOn	EventMuteOn
TListenDisconnect	EventListenDisconnected
TListenReconnect	EventListenReconnected
TSetDNDOn	EventDNDOn
TSetDNDOff	EventDNDOff
TSetMessageWaitingOn	EventMessageWaitingOn
TSetMessageWaitingOff	EventMessageWaitingOff
Query Requests	
TQueryAddress [a]	EventAddressInfo
TQueryCall [a]	EventPartyInfo
TQueryLocation [a]	EventLocationInfo
TQueryServer [a]	EventServerInfo
TQuerySwitch [a]	EventSwitchInfo
User-Data Requests	
TAttachUserData	EventAttachedDataChanged
TUpdateUserData	EventAttachedDataChanged
TDeleteUserData	EventAttachedDataChanged
TDeleteAllUserData	EventAttachedDataChanged
ISCC Requests	
TGetAccessNumber [b]	EventAnswerAccessNumber
TCancelReqGetAccessNumber	EventReqAccessNumberCanceled
Special Requests	

Request	Event
TReserveAgent	EventAgentReserved
TSendUserEvent	EventACK
TSendEvent	EventACK
TSendEventEx	EventACK
TSetCallAttributes	EventCallInfoChanged
TPrivateService	EventPrivateInfo or EventAck

a. Only the requestor will receive a notification of the event associated with this request.
b. Since this feature request may be made across locations in a multi-site environment, if the location attribute of the request contains a value relating to any location other than the local site, except when the response to this request is `EventError`, there will be a second event response that contains the same reference ID as the first event. This second event will be either `EventRemoteConnectionSuccess` or `EventRemoteConnectionFailed`. See Extensions for more information on data passed in multi-site environments.

Reliability

Indicates uncertainty with respect to the *reliability* of an event. For more information, see `Reliability` in your API reference.

RouteType

The type of routing to be applied to the telephony object in question. For more information, refer to the type `RouteType` in your API reference.

XRouteType

The type of routing between T-Servers to be applied to the telephony object in question. For more information, refer to the type `XRouteType` in your API reference.

Server

A local server handle to the T-Server in question. In other words, a unique identifier assigned by T-Library to the connection between a client and T-Server. For more information, refer to the type `TServer` in your API reference.

ServerRole

Specifies the Role of T-Server. For more information, refer to the type `ServerRole` in your API reference.

ServerVersion

The version (release) number of the running T-Server, for example, `7.2.000.02`.

SessionID

A unique session identifier generated by T-Server.

SubscriptionID

A unique subscription identifier generated by T-Server on the creation of a new transaction monitoring subscription.

ThirdPartyDN

The directory number of the third most significant telephony object (except an ACD group or trunk group) with respect to the event in question. The application does not have to be registered to this directory number to receive the event in question.

ThirdPartyDNRole

The role of the telephony object specified by `ThirdPartyDN` in the event in question. For more information, refer to the type `DNRole` in your API reference.

ThirdPartyQueue

The directory number of the third most significant ACD group with respect to the event in question.

ThirdPartyTrunk

The identifier of the third most significant trunk group with respect to the event in question.

ThisDN

The directory number of the most significant telephony object (except an ACD group or trunk group) with respect to the event in question. The application must be registered to this directory number to receive the event in question.

ThisDNRole

The role of the telephony object specified by `ThisDN` in the event in question. For more information, refer to the type `DNRole` in your API reference.

ThisQueue

The directory number of the most significant ACD group with respect to the event in question.

ThisTrunk

The identifier of the most significant trunk with respect to the event in question.

time

The structure specifies event generation time that is expressed in elapsed seconds and microseconds since `00:00 GMT, January 1, 1970` (zero hour). For more information, refer to the type `Time` in your API reference.

TreatmentType

The type of treatment to be applied to the telephony object in question. For more information, refer to the type `TreatmentType` in your API reference.

UserData

Specifies the pointer to the call-related user data. For more information about user data, refer to the `KVList` section of your API reference.

WorkMode

This attribute indicates the agent/supervisor-related current work mode. For more information, refer to the type `AgentWorkMode` in your API reference.

XReferenceID

The reference number of a `TGetAccessNumber()` function that is called by an application.

TEvent Structure

This section describes the syntax of the TEvent structure. For information on types of attributes and possible values, see a full description of this structure in your API reference.

Important

Although listed here, certain components of the TEvent structure are reserved for internal use only.

```
typedef struct TEvent_tag {
        enum TMessageType                         Event;
        TServer                                 Server;
        int                                         ReferenceID;
        char                                         *HomeLocation;
        char                                         *CustomerID;
        TConnectionID                                 ConnID;
        TConnectionID                                 PreviousConnID;
        TCallID                                 CallID;
        int                                         NodeID;
        TCallID                                 NetworkCallID;
        int                                         NetworkNodeID;
        TCallHistoryInfo                         CallHistory;
        TCallType                                 CallType;
        TCallState                                 CallState;
        TAgentID                                 AgentID;
        TAgentWorkMode                                 WorkMode;
        long                                         ErrorCode;
        char                                         *ErrorMessage;
        TFile                                         FileHandle;
        char                                         *CollectedDigits;
        char                                         LastCollectedDigit;
        TDirectoryNumber                         ThisDN;
        TDirectoryNumber                         ThisQueue;
        unsigned long                                 ThisTrunk;
        TDNRole                                 ThisDNRole;
        TDirectoryNumber                         OtherDN;
        TDirectoryNumber                         OtherQueue;
        unsigned long                                 OtherTrunk;
        TDNRole                                 OtherDNRole;
        TDirectoryNumber                         ThirdPartyDN;
        TDirectoryNumber                         ThirdPartyQueue;
        unsigned long                                 ThirdPartyTrunk;
        TDNRole                                 ThirdPartyDNRole;
        TDirectoryNumber                         DNIS;
        TDirectoryNumber                         ANI;
        TAddressInfoType                         InfoType;
        TAddressInfoStatus                         InfoStatus;
        TTreatmentType                                 TreatmentType;
        TRouteType                                 RouteType;
        char                                         *ServerVersion;
        TServerRole                                 ServerRole;
        TMask                                         Capabilities;
        TKVList                                 *UserData;
        TKVList                                *Reasons;
        TKVList                                *Extensions;
        TTimeStamp                                Time;
        void                                        *RawData;
        TDirectoryNumber                        AccessNumber;
        TXRouteType                                XRouteType;
        TReferenceID                                XReferenceID;
        TKVList                                *TreatmentParameters;
        char                                        *Place;
        int                                        Timeout;
        TMediaType                                MediaType;
        TLocationInfoType                        LocationInfo;
        TMonitorNextCallType                        MonitorNextCallType;
        TPrivateMsgType                        PrivateEvent;
        /* Application data (set by TSetApplicationData) */
        void *ApplicationData;
} TEvent;
```

T-Library Call-Based Notifications

This section describes the events generated when T-Server notifies a client of call-based activity. Each event listed here is identified with a description, the contents of the event (presented in table format as a list of the attributes associated with it), and an example of where the event is likely to be encountered during a call flow. (The format of this part of the section is the same as the general events section, T-Library Events.)

Important

The information in this T-Library Call-Based Notifications section replaces a now deprecated solution that used existing DN-based functions to obtain call-based notifications. That former solution required that T-Server clients register for abstract DNs in a Genesys implementation. In order to indicate that an abstract DN was notifying or was being referred to by a parameter of a function, a double colon was used after certain event attributes or parameters, namely `dn` and `location`, as in `AttributeThisDN=<location>::`.

This section has the following subsections:

- Call Definition
- T-Library Call-Based Events
- Multi-Site Call Scenarios

Call Definition

A call is a temporary association among several telephony objects (or between a telephony object and a network entity) that is established by the use of telephony network capabilities. This association may have from zero to many parties. A call is a stateless object that has a Universally Unique ID (UUID) attribute, a Connection ID (comparable to the DN-based attribute of the same name used in event reporting), a type, a number of optional attributes, and a list of parties.

> **Important**
>
> Consultation calls not only have references to the active call, but also to the main (original) call. In multi-site environments, related calls are linked to each other my means of Inter-Site Links (IS-Links). (See Multi-Site Call Scenarios for details.) Except for these two cases of additional references, all calls are processed and reported on independently of one another.

Call Attributes

`CallUUID` (string)—UUID of the call.
`ISLinkList`—List of links to call instances distributed across remote sites.
`ConnID` (conn-id)—Connection ID of the call.
`CallType` (int)—Call type (Internal/Inbound/Outbound/Consult).
`OrigCallUUID`—UUID of the main call (for consult calls only).

> **Important**
>
> The following additional attributes for calls have the same definitions as their DN-based analogs. See individual listings under Event Attributes.

`CallID`
`NodeID` (int)
`NetworkCallID` (ulong64)
`NetworkNodeID` (int)
`ANI` (string)
`DNIS` (string)
`UserData` (kv-list)

T-Library Call-Based Events

This section describes the call-based events and how they differ from their DN-based counterparts.

Event contents are presented as the collection of attributes associated with each event, as well as an indication of that attribute's type. For the purposes of this section, type has one of two values: `Mandatory` or `Optional`. Here type refers to the presence of the attribute at the time of the generation of its event, and not to a characteristic of the attribute itself.

Attribute Type

- Mandatory—Indicates that this attribute is always present when its associated event occurs.
- Optional—Indicates that this attribute may or may not be present when the associated event occurs.

List of T-Library Call-Based Events

There are only three notifications available for calls: EventCallCreated, EventCallDataChanged, and EventCallDeleted. Each call-based event may have a number of attributes, and, except for a call's `CallUUID`, all call attributes (including `ConnID` and `ISLinkList`) may have values that change. For Hot Standby redundant environments, `CallUUID` is replicated from the primary to the backup T-Server.

> **Important**
> T-Server may report several changes to a call in one event, even if the changes are caused by different CTI messages.

EventCallCreated

Description

Indicates that a call has been created in T-Server.

Contents

Event Attribute	Type
CallUUID	Mandatory
ISLinkList [a]	Optional

Event Attribute	Type
ConnID	Mandatory
CallType	Optional
OrigCallUUID	Optional
CallID	Optional
NodeID	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
ANI	Optional
DNIS	Optional
UserData	Optional
CustomerID	Optional
DialedNumber [b]	Optional

a. Applies to resynchronization cases only (T-Server with clients upon initial connect or reconnect at HA switchover). Otherwise, this value is typically unknown at call creation.
b. Applies to outgoing calls only. This optional attribute (string) contains the number dialed, if known when the call is created (which would be possible only if it had been created at a client's request).

EventCallDataChanged

Description

Indicates that some of a given call's properties have changed.

Contents

Important

Unlike DN-based notifications, `EventCallDataChanged` includes as attributes only those call attributes that have changed. (Except for `CallUUID` and `ConnID`, unchanged attributes are not present in the event, including `ISLinkList`; in the event that user data is deleted from the call, the `UserData` attribute contains an empty `TKVList`.)

Event Attribute	Type
CallUUID	Mandatory
ISLinkList	Optional
ConnID	Mandatory

Event Attribute	Type
CallType	Optional
OrigCallUUID	Optional
CallID	Optional
NodeID	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
ANI	Optional
DNIS	Optional
UserData	Optional
CustomerID	Optional
CtrlParty [a]	Optional

a. A reference to the party that caused the change.

EventCallDeleted

Description

Indicates that the call has been deleted.

Contents

Event Attribute	Type
CallUUID	Mandatory
ConnID	Mandatory
Cause [a]	Optional
RefCallUUID [b]	Optional
RefConnID [c]	Optional
CtrlParty [d]	Optional

a. An integer indicating the cause of the deletion, for instance, a transfer or conference.
b. `CallUUID` for the call to which this one was merged, as in the case of a transfer, conference, or merge.
c. The `RefConnID` attribute can only be present if the cause of the call deletion is a transfer, conference, or merge.
d. A reference to the party that initiated the disconnection of the call.

EventReleased — DEPRECATED

Description

The telephony object specified by `ConnID` has been deleted from T-Server memory.

Important

T-Server delivers this version of `EventReleased` when the DN referred to by the parameter `ThisDN` is an abstract DN. (Here the attribute `ThisDN` uses the name of the T-Server application as listed in the Configuration Layer, with a double colon added, or the name of the switch, also with a double colon added.) Clients receive this event each time a call is released on each of the abstract DNs for which they are registered.

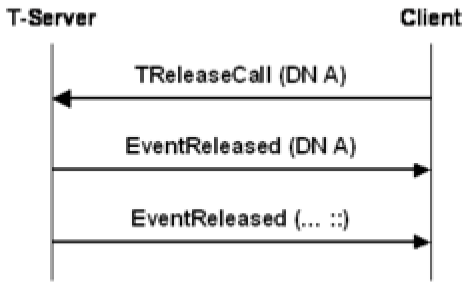
EventReleased Contents—DEPRECATED

Event Attribute	Type
ANI	Optional
CallHistory	Optional
CallID	Mandatory
CallState	Mandatory
CallType	Mandatory
ConnID	Mandatory
RefConnID	Optional [a]
CollectedDigits	Optional
CustomerID	Optional
DNIS	Optional
Event	Mandatory
Extensions	Optional
NetworkCallID	Optional
NetworkNodeID	Optional
PreviousConnID [b]	Optional
Reasons	Optional
Server	Mandatory
ThisDN [c]	Mandatory
time	Mandatory
TransferredNetworkCallID	Optional
TransferredNetworkNodeID	Optional

Event Attribute	Type
UserData	Optional
Reliability	Mandatory

a. This attribute is mandatory for call `CallStateTransferred`, `CallStateConferenced`, and `CallStateRemoteRelease`.
b. The attribute must appear if `CallType` is `Consult`.
c. The attribute `ThisDN` uses the name of the T-Server application as listed in the Configuration Layer or the name of the switch. In both cases, the indication that `EventReleased` is based on an abstract DN is through the use of the double colon in the notification, as in `AttributeThisDN=<location>::`, where `location` is the name of the T-Server or switch.

Example



EventReleased Feature Example—DEPRECATED

For more information, refer to Call Models and Flows.

Extensions in EventAddressInfo and EventRegistered for Abstract DNs—DEPRECATED

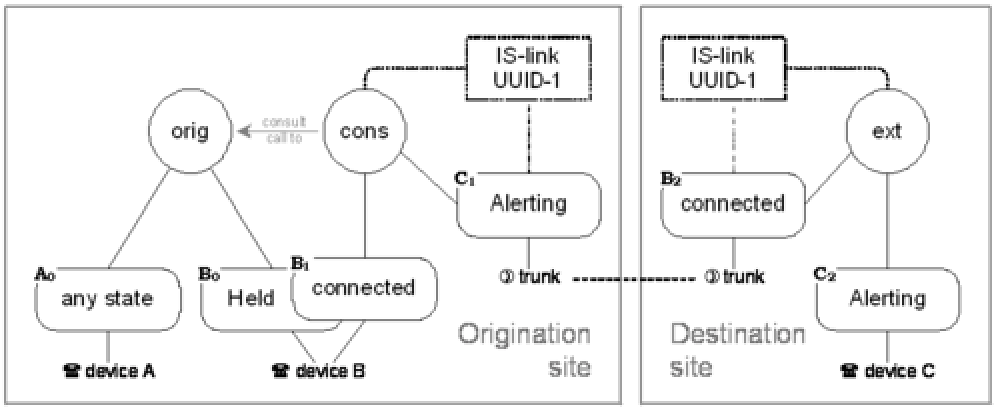
InfoStatus	Attribute Extensions		
Key	**Value Type**	**Value Description**	
InfoType=AddressInfoDNStatus			
NumberOfCalls	conn-%d	string	The ConnectionID of a call (converted into a string)
	ct-%d	integer	The CallType for the corresponding ConnectionID

Multi-Site Call Scenarios

A basic principle for multi-site reporting is that T-Server does not try to support the same model for multi-site calls as it does for those that are local. Instead, it is the client's responsibility to merge call information into a single unit based on the call's relationship information provided by T-Server. To help with multi-site call reporting, Genesys has introduced the Inter-Site Link (IS-Link) with the following properties:

- Each IS-Link has a UUID, assigned at the time of creation and reported in `ISLinkList.`
- For Hot Standby redundant environments, IS-Links on each side are also replicated from the primary to the backup T-Server.
- At any moment, IS-Link may be associated with no more than two calls (located on different T-Servers).
    - IS-Link may represent a communication channel, for instance, a voice channel between calls in different switching domains.
    - Alternatively, IS-Link may associate two calls that continue each other, connected together through association with the same external participant, for instance, an incoming call overflowed or re-routed by an external network to another switching domain.
- On each T-Server the IS-Link–associated call is reported in the context of the `CallUUID,` independent of the other T-Server.
- The IS-Link call association does not depend on any T-Server options. In the Multi-Site Call Transfer diagram, for instance, the link is always connected to the call labelled `cons,` even if T-Server is instructed to populate the `ConnID/UserData` of the call labelled `orig` for the remote site.
- An IS-Link may be re-attached to a different call on the same T-Server, but only as a result of a merge operation. This action is not reported explicitly, but assumed from `EventCallDeleted,` with a `RefCallUUID` attribute specification.
- A call may have more than one associated IS-Link.
- In some scenarios when there is an overflow from a queue, an IS-Link may be attached to a call after the call is reported as deleted.

The following diagram illustrates a case of three related calls, of which only two are linked together.



Multi-Site Call Transfer

The association between an IS-Link and a call is reported in `EventCallDataChanged`, which is distributed to call-monitoring clients. Often, the same event may also report a change in `ConnID` and `UserData` as the result of multi-site synchronization.

> **Important**
> In most cases, IS-Link is attached before the first DN-based event is sent. (That is, before `EventRinging` and `EventRouteRequest`.) The exception is with events for trunks that have trunk monitoring. T-Server clients should be capable of processing later updates.

Simple Multi-Site Call

In this scenario, events for which are described in the following table, DN A on Site 1 calls DN B on Site 2.

Origination (Site 1)			**Destination (Site 2)**	
DN-Based Events	**Call-Based Notifications**	**DN-Based Events**	**Call-Based Notifications**	
A: TMakeCall to B (Site 2)				
	EventCallCreated CallUUID **UUID-X** ConnID **x**			
EventDialing CallUUID **UUID-X** ConnID **c** (**x** replaced by ISCC) ThisDN **A** OtherDN **B** [a]	**EventCallDataChanged** CallUUID **UUID-X** ConnID **c** ISLinkList **UUID-L@'site2'**			
EventNetworkReached (Optional) CallUUID **UUID-X** ConnID **c** ThisDN **A** OtherDN **B** [a]				**EventCallCreated** CallUUID **UUID-Y** ConnID **y**
			EventRinging CallUUID **UUID-Y** ConnID **c** (**y** replaced by ISCC) ThisDN **B** OtherDN **A** [a]	**EventCallData-Changed** CallUUID **UUID-Y** ConnID **c** ISLinkList **UUID-L@'site1'**
			B: TAnswerCall()	

Origination (Site 1)			Destination (Site 2)	
EventEstablished CallUUID **UUID-X** ConnID **c** ThisDN **A** OtherDN **B** [a]			**EventEstablished** CallUUID **UUID-Y** ConnID **c** ThisDN **B** OtherDN **A** [a]	

Multi-Site Call Transfer

In this scenario, events for which are described in the following table, DN B on Site 1 initiates a transfer for an existing call to DN C on Site 2.

Origination (Site 1)			Destination (Site 2)	
DN-Based Events	**Call-Based Notifications**	**DN-Based Events**	**Call-Based Notifications**	
B is in conversation with A. **B: TInitiateTransfer to C (Site 2)**				
	EventCallCreated CallUUID **UUID-X** OriginCallUUID **UUID-W** ConnID **c**			
EventDialing CallUUID **UUID-X** ConnID **cons** (**c** replaced by ISCC) TransferConnID **orig** ThisDN **B** OtherDN **C**	**EventCallDataChanged** CallUUID **UUID-X** ConnID **cons** ISLinkList **UUID-L@'site2'**			
EventNetworkReached (Optional) CallUUID **UUID-X** ConnID **cons** TransferConnID **orig** ThisDN **B** OtherDN **C**				**EventCallCreated** CallUUID **UUID-Y** ConnID **y**
			EventRinging CallUUID **UUID-Y** ConnID **ext** [a] (**y** replaced by ISCC) ThisDN **C** OtherDN **B**	**EventCallData-Changed** CallUUID **UUID-Y** ConnID **ext** [a] ISLinkList **UUID-L@'site1',** (optional: uuid-2B)
			B: TAnswerCall()	

Origination (Site 1)			Destination (Site 2)	
EventEstablished CallUUID **UUID-X** ConnID **cons** ThisDN **B** OtherDN **C**			**EventEstablished** CallUUID **UUID-Y** ConnID **ext** [a] ThisDN **C** OtherDN **B**	

a. For compatibility with existing solutions, `ConnID` is assigned at the remote site based on the value of the `use-data-from` option at the origination T-Server. If there exists a call with a duplicate `ConnID`, a new, unique ID is generated (and, for DN-based reporting purposes, the reference to the other call is passed in the `LastTransferConnID` attribute).

Transfer/Conference Completion

At the completion of a transfer or conference, the origination T-Server distributes the same events as in a standard single-site model. The destination T-Server may not report the completion of the transfer or conference at all, or may report it and even report on the call context change, for instance, if EPP is in effect. Transfer completion might be reported as in the table below.

For transfer completion, no explicit reporting of changes in call relations is required. `EventCallDeleted` with `RefCallUUID` implies a move of all previously attached IS-Links from call `UUID-X` to `UUID-W`.

Multi-Site Call Transfer Completion

Origination (Site 1)			Destination (Site 2)	
DN-Based Events	**Call-Based Notifications**	**DN-Based Events**	**Call-Based Notifications**	
EventReleased CallUUID **UUID-W** ConnID **orig** ThisDN **B** OtherDN **A** **EventReleased** CallUUID **UUID-X** ConnID **cons** TransferConnID **orig** ThisDN **B** OtherDN **C** **EventPartyChanged** CallUUID **UUID-W** ConnID **orig** PreviousConnID **orig** ThisDN **A** OtherDN **C**				
	EventCallDeleted CallUUID UUID-X RefCallUUID **UUID-W** Cause **Transfer**		**EventPartyChanged** (Optional) CallUUID **UUID**-Y	**EventCallDataChanged** (Optional) CallUUID **UUID-Y**

Origination (Site 1)			Destination (Site 2)	
	CtrlParty **B**		ThisDN **C** ConnID **orig** PreviousConnID **ext**	ConnID **orig**

Attaching the IS-Link Post Mortem

The following three scenarios (Simple Call Overflow, Overflow On the Intermediate Switch, and Mute Transfer With Overflow) describe instances where the IS-Link is assigned in non-standard ways, after the call has been deleted.

Simple Call Overflow

In the case of call overflow, even for a simple scenario, T-Server may discover the relation between two calls only after overflowed call has already been deleted.

Origination (Site 1)			Destination (Site 2)	
DN-Based Events	**Call-Based Notifications**	**DN-Based Events**	**Call-Based Notifications**	
EventQueued CallUUID **UUID-X** ConnID **loc** ThisDN **B** OtherDN **A**	**EventCallCreated** CallUUID **UUID-X** ConnID **loc**			
EventDiverted CallUUID **UUID-X** ConnID **loc** ThisDN **B** OtherDN **A**	**EventCallDeleted** CallUUID **UUID-X**			
	EventCallDataChanged CallUUID **UUID-X** ISLinkList **UUID-L@‘site2'**		**EventQueued** CallUUID **UUID-Y** ConnID **rem** (**y** replaced by ISCC) ThisDN **C** OtherDN **A**	**EventCallCreated** CallUUID **UUID-Y** ConnID **y** **EventCallData-Changed** CallUUID **UUID-Y** ConnID **rem** ISLinkList **UUID-L@‘site1'**

Overflow On the Intermediate Switch

In the case of a call being overflowed on the intermediate switch (distinct from both the original and destination), or overflowed two or more times, call information may be deleted at the transit site, even if that information still exists at the end site. The reporting is similar to the case of the simple overflow, and is shown below.

Origination (Site 1)		**Transit (Site 2)**		**Destination (Site 3)**
Call-Based Notifications	**Call-Based Notifications**	**Call-Based Notifications**		
EventCallCreated CallUUID **UUID-X** ConnID **conn-1**		**EventCallCreated** CallUUID **UUID-Y** ConnID **conn-2** **EventCallDeleted** CallUUID **UUID-Y**		
EventCallDataChanged CallUUID **UUID-X** ConnID **rem1** ISLinkList **UUID-L1@‘site2'**		**EventCallDataChanged** CallUUID **UUID-Y** ISLinkList **UUID-L1@‘site1'**		**EventCallCreated** CallUUID **UUID-Z** ConnID **conn-3**
		EventCallDataChanged CallUUID **UUID-Y** ISLinkList **UUID-L1@‘site1'** **UUID-L2@‘site3'**		**EventCallDataChanged** CallUUID **UUID-Z** ConnID **rem3** ISLinkList **UUID-L2@‘site2'**

Mute Transfer With Overflow

In case of a mute transfer to a remote site, in combination with a call overflow, the relationship between the two calls may be discovered after the active call has been merged into the main call.

Origination (Site 1)			**Destination (Site 2)**	
DN-Based Events	**Call-Based Notifications**	**DN-Based Events**	**Call-Based Notifications**	
...	...			
EventPartyChanged CallUUID **UUID-W** ConnID **orig** PreviousConnID **cons** ThisDN **A**				
	EventCallDeleted CallUUID **UUID-X** RefCallUUID **UUID-W** Cause **Transfer**			
				EventCallCreated

Origination (Site 1)			**Destination (Site 2)**	
				CallUUID **UUID-Y** ConnID **y**
	EventCallDataChanged CallUUID **UUID-X** ISLinkList **UUID-L@‘site2'**		**EventRinging** CallUUID **UUID-Y** ConnID **ext** (**y** replaced by ISCC) ThisDN **C** OtherDN **B**	**EventCallData-Changed** CallUUID **UUID-Y** ConnID **ext** ISLinkList **UUID-L@‘site1'**

Client-Side IS-Link Processing

Genesys recommends taking into consideration the following principles when developing for client-side support of multi-site call-linkage discovery.

- If the same IS-Link is listed for two calls, the calls are linked together.
- An IS-Link relation is transitive: If A is linked to B, and B linked to C, then A is also linked to C.
- In order to accommodate overflow scenarios and possible network delays, you should preserve call information, even after `EventCallDeleted`, for a short time before purging it.
- In order to accommodate transitive call linkage, call information should be preserved as long as there are two or more active IS-Link associations with a call for which information would otherwise be purged. (For instance, avoid unlinking A and C by deleting B's call information.)
- Static storage of IS-Links can be presented in a table indexed by `ISLinkUUID`. In such a table, each IS-Link record provides placeholders for two calls (UUIDs) and two sites.
- For situations where only a subset of sites is visible to a client or an external routing request proves unsuccessful, the IS-Link record may remain incomplete (refer to one call only). These types of records will be discarded when the single call is purged.
- For dynamic IS-Link storage when one call is merged with another, all IS-Links should be moved to (or applied towards) the remaining call.
- An IS-Link record expires when either of two calls is purged.

ISCC Transaction Monitoring

ISCC Transaction monitoring is a feature for working with multi-site environments. ISCC is an internal T-Server component responsible for T-Server's multi-site operations such as multi-site call transfer, inter-site call linkage, call overflow, and other, possibly non-call–related, multi-site operations. Internally, those operations are called *transactions*.

This section describes the interfaces that allow you to work with ISCC Transaction Monitoring: the `TTransactionMonitoring()` function, the `TSubscriptionOperationType` enumeration, and `EventTransactionStatus.` Use these to subscribe and work with this feature. In particular, the key-value pairs of the `Extensions` attribute returned in `EventTransactionStatus` expose details of call-transfer availability (the current state of ISCC transactions), such as delays and losses, in the multi-site environment.

The content of this section represents the implementation of the transaction model exposed as an extension of the existing T-Library SDK.

This section has the following subsections:

- The Subscription Type
- Subscribing to Transaction Monitoring
- The Transaction Monitoring Event
- Transaction Monitoring States
- Transaction Monitoring Elements

The Subscription Type

There is one enumeration that relates to transaction monitoring, `TSubscriptionOperationType`.

TSubscriptionOperationType

This enumeration defines an operation on a subscription to Transaction Monitoring Feature notifications.

Syntax

```
typedef enum {
        SubscriptionStart,
        SubscriptionStop,
        SubscriptionModify
} TSubscriptionOperationType;
```

Values

- `SubscriptionStart`—Request to create a new subscription.
- `SubscriptionStop`—Request to stop an existing subscription.
- `SubscriptionModify`—Request to modify the rules of an existing subscription.

Subscribing to Transaction Monitoring

To initiate transaction monitoring, your code must call the following function.

TTransactionMonitoring

This function requests a T-Server to start, stop, or modify a subscription to the Transaction Monitoring Feature notifications. If a request is successful (EventACK), EventTransactionStatus (EventTransactionStatus) is distributed to the requestor.

Syntax

```
int TTransactionMonitoring (
        TServer                         tserver_handle,
        TSubscriptionOperationType         subscription_operation,
        const char                         * subscription_identifier,
        const TKVList                         * subscription_rules
);
```

Parameters

- tserver_handle—Local server handle to the T-Server in question.
- subscription_operation—This parameter represents an operation on a subscription to Transaction Monitoring Feature notifications. Its value is one of TSubscriptionOperationType. A value of this parameter should be provided by the T-Server client.
- subscription_identifier—This parameter represents a subscription identifier. A value should be NULL for the operation SubscriptionStart. For other operations its value should be taken from the message EventACK. It is generated by a T-Server on a creation of a new subscription.
- subscription_rules—This parameter contains subscription rules. A value of this parameter should be provided by the T-Server client. Its value should be NULL for the operation SubscriptionStop. See Subscription Rules for details.

Return Values

The Transaction Monitoring Feature return value is a standard return value for the T-Library API:

- >0—Reference number (ReferenceID) assigned to this request by the T-Library API.
- <0—Error condition.

Subscription Rules

Subscription rules are provided by T-Server clients and passed as part of a `notify` key-value pair in the `Extensions` attribute for `RequestTransactionMonitoring`. They are represented as a key-value list.

Every subscription rule key-value pair contains a notification mode for one element. The key represents the name of the element. The value represents the requested notification mode. If no key has as its name a given element, then that element's default notification mode is used. See Transaction Monitoring Elements for the list of all default notification modes.

Example

The following is an example `RequestTransactionMonitoring` with subscription rules included:

```
AttributeReferenceID reference-id-1
AttributeSubscriptionOperation SubscriptionStart
AttributeExtensions
        notify (list)
                class.name always
                object.local-id always
                iscc.transaction.state always
                iscc.direction.role once
                iscc.is-link-creation never
```

The local attribute `object.local-id` represents a unique identifier of the Local Transaction Instance. The value of this attribute is different from the value of the Global Identifier attribute.

The Transaction Monitoring Event

You can expect to receive `EventTransactionStatus` once you have successfully subscribed to transaction monitoring. Information about your subscription request is also available from `EventACK`, `EventError`, and `EventServerInfo`. For descriptions of these general T-Library events and their transaction monitoring specifics, see EventACK, EventError, and EventServerInfo. For descriptions of event attributes generally, see Event Attributes.

EventTransactionStatus

Description

This message is an implementation of Transaction Monitoring Feature notification. This notification is associated with one of the subscriptions requested by the T-Server client. The key-value pairs of the `Extensions` attribute expose details of call-transfer availability (the current state of ISCC transactions), such as delays and losses, in the multi-site environment.

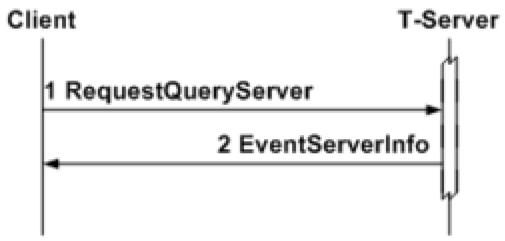
Contents

Event Attribute	Type
SubscriptionID [a]	Mandatory
Extensions	Mandatory

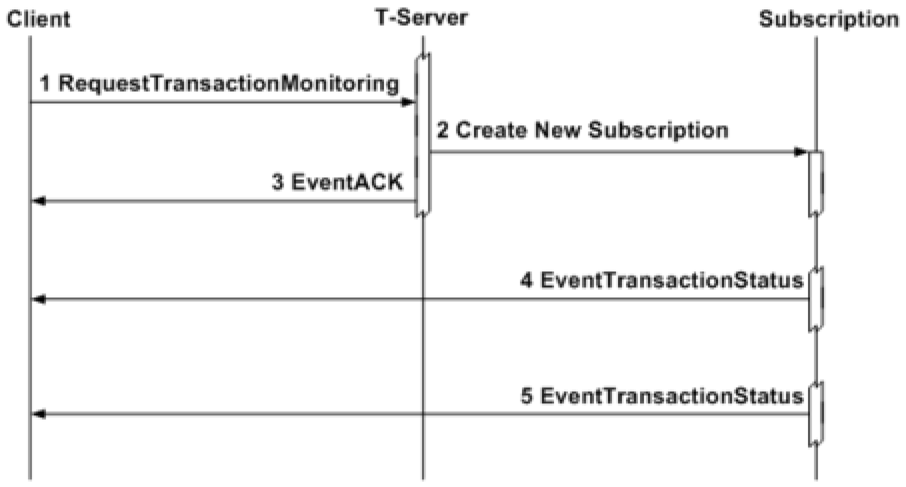
a. For a description of this attribute, see SubscriptionID in the general Event Attributes descriptions.

Examples

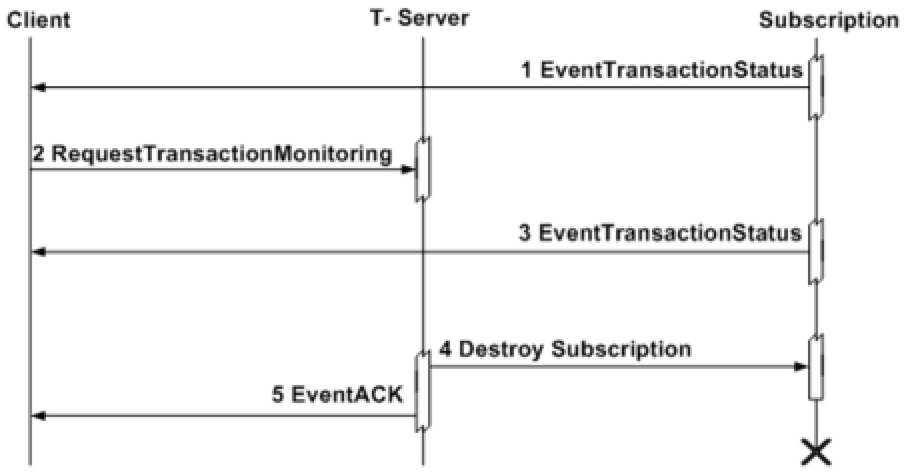
The following examples demonstrate the main requests and expected responses for cases related to transaction monitoring.



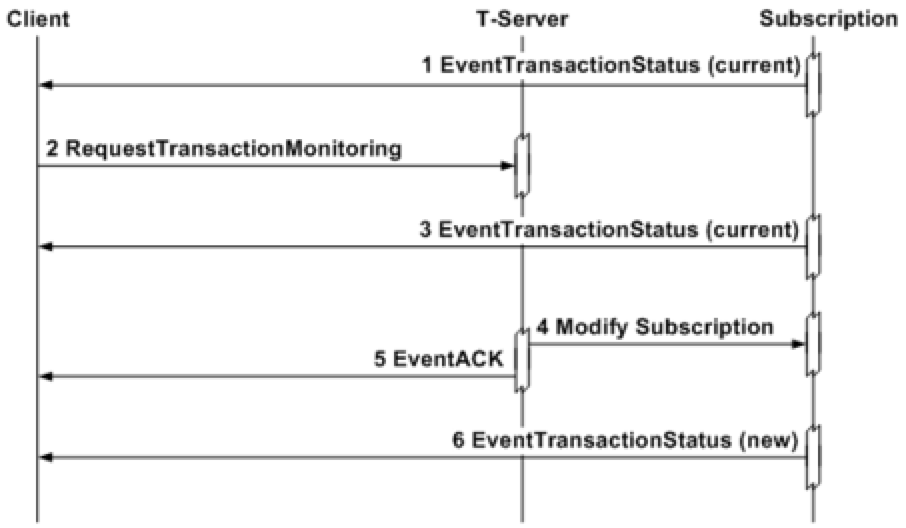
Transaction Monitoring Feature: Query Server



Transaction Monitoring Feature: Subscribe



Transaction Monitoring Feature: Unsubscribe

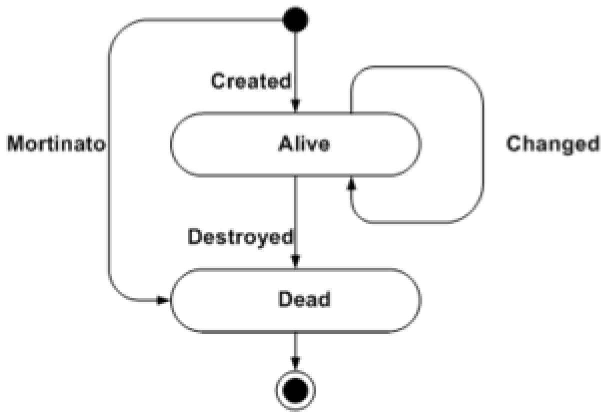

Client
T-Server
Subscription
1 EventTransactionStatus (current)
2 RequestTransactionMonitoring
3 EventTransactionStatus (current)
4 Modify Subscription
5 EventACK
6 EventTransactionStatus (new)

Transaction Monitoring Feature: Modify Subscription

Transaction Monitoring States

This section presents models (including substates and transitions) of the states related to transaction monitoring: the Object, Abstract Transaction, IS Link Creation Feature, Call Operation Feature, and Resource Feature states.

Object State



Object State

Transitions

- `created`—For an object that has just been created; the first transition for every object (from initial to alive).
- `changed`—An object's attributes have been changed (from alive to alive).
- `destroyed`—An object has just been destroyed; the last transition for every object (from alive to dead).
- `mortinato`—An object has been created and destroyed at one time; the last transition for every object (from initial to dead).

Appearance

The Object component is present with every object instance, from the initial to the final transition. All attributes of an object component are present with every object instance at all times.

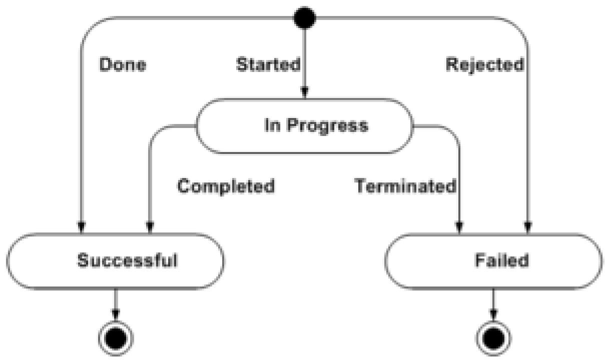
If the object instance has exactly one transition, for instance, `iscc.transaction.rejected` or `iscc.transaction.done`, the object component of such an object instance has one transition as well. In this case the transition is `object.mortinato`.

If the object instance has two transitions, for instance, `iscc.transaction.started` and `iscc.transaction.completed`, the object component of the object instance has two transitions: `object.created` and then `object.destroyed`.

If the object instance has more then two transitions, the object component of this object instance is present for every notification event. The first notification event has a transition of `object.created`.

The last notification event has a transition `object.destroyed`. Notification events that occur between those two have a transition of `object.changed`.

Abstract Transaction State



Abstract Transaction State

SubStates

- **in-progress**—The transaction instance is in progress.
- **successful**—The transaction instance has successfully completed.
- **failed**—The transaction instance has abnormally terminated.

Transitions

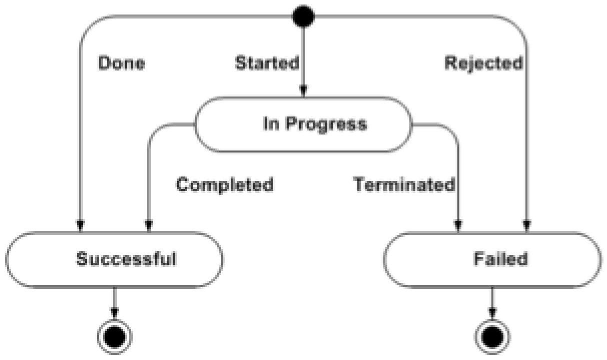
- `started`—Transaction instance has started (from initial to in-progress).
- `completed`—Transaction instance has completed successfully (from in-progress to successful).
- `terminated`—Transaction instance has completed abnormally (from in-progress to failed).
- done—Transaction instance has started and has completed successfully at the same time (from initial to successful). This transition might be considered as two consecutive transitions—`started` and `completed`—occurring without an intermediate delay. Such a transition might happen when the initial conditions for this transaction instance are met (all the required resources are available), and the transaction can be completed at this same time.
- `rejected`—Transaction has started and has completed abnormally at the same time (from initial to failed). You could consider this transition as two consecutive transitions—`started` and

terminated—occurring without an intermediate delay. Such a transition might happen when the initial conditions for this transaction are not met (a required resource is permanently unavailable), or the transaction can never succeed.

Appearance

The Abstract Transaction component is present with every object instance, from the initial to the final transition. All attributes of an abstract transaction component are present with every object instance and at all times.

IS Link Creation Feature State



IS Link Creation Feature State

SubStates

- **in-progress**—IS Link creation is in progress.
- **successful**—IS Link creation has successfully completed.
- **failed**—IS Link creation has terminated abnormally.

Transitions

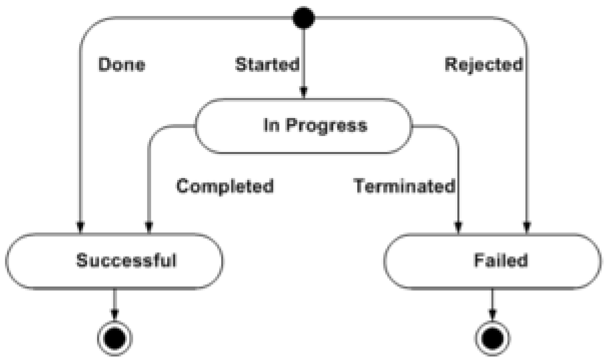
- `started`—IS Link creation has started (from initial to in-progress).
- `completed`—IS Link creation has completed successfully (from in-progress to successful).
- `terminated`—IS Link creation has completed abnormally (from in-progress to failed).
- `done`—IS Link creation has started and has completed successfully at the same time (from initial to successful). This transition might be considered as two consecutive transitions—`started` and `completed`—occurring without an intermediate delay.
- `rejected`—IS Link creation has started and has completed abnormally at the same time (from initial to failed). You could consider this transition as two consecutive transitions—`started` and

terminated—occurring without an intermediate delay.

Appearance

The IS Link Creation feature component works asynchronously with respect to the Call Operation and Resource features. The IS Link Creation feature component starts when all checks for transaction validity have successfully passed. It ends when the transaction instance ends. Attributes of this component are present with the transaction instance from the time the IS Link Creation feature component starts and till it ends.

Call Operation Feature State



Call Operation Feature State

SubStates

- **in-progress**—Call operation is in progress.
- **successful**—Call operation has successfully completed.
- **failed**—Call operation has abnormally terminated.

Transitions

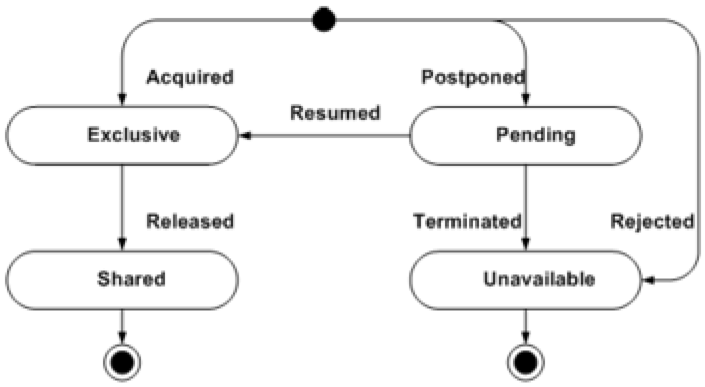
- `started`—Call operation has started (from initial to in-progress).
- `completed`—Call operation has successfully completed (from in-progress to successful).
- `terminated`—Call operation has completed abnormally (from in-progress to failed).
- `done`—Call operation has started and completed at the same time (from initial to successful). This transition might be considered as two consecutive transitions—`started` and `completed`—occurring without an intermediate delay.
- `rejected`—Call operation has started and has completed abnormally at the same time (from initial to failed). You could consider this transition as two consecutive transitions—`started` and

`terminated`—occurring without an intermediate delay.

Appearance

The Call Operation feature component works asynchronously with respect to the IS Link Creation and Resource features. The Call Operation feature component starts when ISCC starts performing a given call operation to fulfill a multi-site operation. Usually, it represents a request/response message pair. The Call Operation feature component ends when the transaction instance ends. (The attribute `iscc.call-operation.call-id` is present when a call related to the call operation is known.)

Resource Feature State



Resource Feature State

SubStates

- **exclusive**—The resource has been acquired in `exclusive` mode.
- **shared**—The resource has been released, and might be acquired by another transaction.
- **pending**—A resource is temporarily unavailable.
- **unavailable**—A resource is permanently unavailable.

Transitions

- `acquired`—The resource has been acquired exclusively (from initial to exclusive).
- `released`—The resource has been released (from exclusive to shared).
- `postponed`—The resource is temporarily unavailable; the transaction has been postponed (from initial to pending).
- `resumed`—The resource is available; the transaction has resumed (from pending to exclusive).

- `rejected`—A resource is unavailable permanently from the outset; the transaction has been rejected (from initial to unavailable).
- `terminated`—A resource became unavailable permanently; the transaction has been terminated (from pending to unavailable).

Appearance

The Resource feature component works asynchronously with respect to the IS Link Creation and Call Operation features. The Resource feature component starts when ISCC tries for first time to acquire a resource required for the multi-site operation. The Resource feature component ends when the transaction instance ends. The attribute `iscc.resource.name` is present from the time when ISCC successfully acquires the corresponding resource.

Transaction Monitoring Elements

The following table contains a full list of transaction monitoring elements and their default notification modes.

Full Set of Transaction Elements

Element Type	Element Name	Default Notification Mode
feature	class	never
attribute	class.id	never
attribute	class.name	once
attribute	class.namespace	never
attribute	class.feature-set	once
feature	object	never
attribute	object.id	always
attribute	object.local-id	never
attribute	object.state	never
state	object.alive	
state	object.dead	
transition	object.created	never
transition	object.changed	never
transition	object.destroyed	never
transition	object.mortinato	never
namespace	iscc	
feature	iscc.transaction	always
attribute	iscc.transaction.state	never
state	iscc.transaction.in-progress	
state	iscc.transaction.successful	
state	iscc.transaction.failed	
transition	iscc.transaction.started	always
transition	iscc.transaction.completed	always
transition	iscc.transaction.terminated	always
transition	iscc.transaction.done	always
transition	iscc.transaction.terminated	always
feature	iscc.direction	always
attribute	iscc.direction.role	once
feature	iscc.is-link-creation	always
attribute	iscc.is-link-creation.is-link-id	once
attribute	iscc.is-link-creation.state	never

Element Type	Element Name	Default Notification Mode
state	iscc.is-link-creation.in-progress	
state	iscc.is-link-creation.successful	
state	iscc.is-link-creation.failed	
transition	iscc.is-link-creation.started	always
transition	iscc.is-link-creation.completed	always
transition	iscc.is-link-creation.terminated	always
transition	iscc.is-link-creation.done	always
transition	iscc.is-link-creation.rejected	always
feature	iscc.call-operation	always
attribute	iscc.call-operation.call-id	on-change
attribute	iscc.call-operation.state	never
state	iscc.call-operation.in-progress	
state	iscc.call-operation.successful	
state	iscc.call-operation.failed	
transition	iscc.call-operation.started	always
transition	iscc.call-operation.completed	always
transition	iscc.call-operation.terminated	always
transition	iscc.call-operation.done	always
transition	iscc.call-operation.rejected	always
feature	iscc.resource	always
attribute	iscc.resource.name	once
attribute	iscc.resource.state	never
state	iscc.resource.exclusive	
state	iscc.resource.shared	
state	iscc.resource.pending	
state	iscc.resource.unavailable	
transition	iscc.resource.acquired	always
transition	iscc.resource.released	always
transition	iscc.resource.postponed	always
transition	iscc.resource.resumed	always
transition	iscc.resource.terminated	always
transition	iscc.resource.rejected	always
class	iscc.transaction-route	always
class	iscc.transaction-direct-callid	always
class	iscc.transaction-reroute	always
class	iscc.transaction-direct-uui	always
class	iscc.transaction-direct-ani	always

Element Type	Element Name	Default Notification Mode
class	iscc.transaction-direct-notoken	always
class	iscc.transaction-dnis-pool	always
class	iscc.transaction-direct-digits	always
class	iscc.transaction-pullback	always
class	iscc.transaction-route-uui	always
class	iscc.transaction-network-callid	always
class	iscc.transaction-cof-callid	always
class	iscc.transaction-cof-ani	always
class	iscc.transaction-cof-network-callid	always

T-Library Unstructured Data

This section describes how T-Server accommodates and allows programmers to work with unstructured data. There are three types of unstructured data supported by T-Server: user data, extensions, and reasons.

This section contains the following subsections:

- User Data
- Extensions
- Reasons

User Data

Transaction-related *user data* is structured as a list of data items described as key-value pairs, where the key stands for a parameter name and the value represents the current value of that parameter. Each key-value pair may contain information about only one parameter, whose value can be an integer, character string, binary type, or unicode.

Warning

Support for unicode is not backward compatible. Unicode should only be used in uniform 7.0 environments (where all clients use the 7.0 release of T-Library). In mixed environments, clients built with earlier releases of T-Library will not be able to decode unicode sent from 7.0 clients and servers.

There is no specific size limitation for the number of key-value pairs or for the size of individual keys or values. However, the total size of the key-value pairs is limited. When in a packed format, the total size of the pairs must not exceed the configured value (default is `16,000` bytes); the configured value has a maximum of `65,535` bytes.

Important

Increasing the configured value above `16,000` bytes, a feature introduced with release 7.0, may degrade performance. Moreover, if the value is set above the `16,000`-byte default, release 6.5 components of a Genesys environment cannot properly process the data passed to them, and may even become unstable.

Arrival, Use, and Manipulation of User Data

User Data can arrive at a client application with any event that contains the `UserData` attribute in its TEvent structure. Client applications should be designed with the ability to view that data. Moreover, since applications may need to create or modify user data and send it to T-Server for processing elsewhere in the system, Genesys provides built-in functions for this purpose.

There are a variety of functions available for the creation and manipulation of user data. (See the Voice Platform SDK API Reference .NET or Java for details on all such available functions.) While these functions do not generate requests of T-Server, they allow client applications to create contact-related data structures understandable for other T-Server clients and to read and modify contact-related information coming from other clients via T-Server. A created or modified data structure will typically be sent to T-Server using the `TAttachUserData()` or `TUpdateUserData()` function (both of which are specified in the `tlibrary.h` header file). Examples of how to use user data functions are also available with the SDK documentation.

The functions for the creation and manipulation of user data fall into three broad groups: reading functions, creation functions, and modification functions. Reading functions are invoked every time an application wants to use contact-related information for its internal purposes. Creation and modification functions are only used by an application to make new data available to other client applications of T-Server.

User Data in Consultation Calls

In addition to creating and modifying user data, it is also important to provide ways for user data to be shared among parties on a consultation call. T-Server provides three methods of handling user data for consultation calls. In the first method, called the *separate method*, user data for the consultation call is attached and stored separately from the user data attached to the original call. In the second, called the *inherited method*, user data attached to the original call is copied to the consultation call at the moment the consultation call is initiated; after that, any changes to the original call's user data do not affect the consultation call's user data, and vice versa. In the third, called the*joint method*, user data attached to either the original or the consultation call is associated with the original call and, yet, can be seen and changed by the parties of both calls.

A consultation call can be initiated with a T-Library request, such as `TInitiateConference()`, `TInitiateTransfer()`, or `TMuteTransfer()`. Use the `Extensions` attribute with the `ConsultUserData` key specified in the request to set the method of handling user data for a consultation call. Extensions for details on this attribute in other contexts. A key-value pair with the `ConsultUserData` key can have the following values:

- `default`
- `separate`
- `inherited`
- `joint`

Default Value

Applying the value `default` to the `ConsultUserData` key causes T-Server to use the current value specified for its `consult-user-data` configuration option. See your specific T-Server Deployment Guide to learn more about this option. If a conference/transfer request does not contain the `ConsultUserData` key, the value specified in the `consult-user-data` option applies. Otherwise, the method of handling user data is based on the value of the `ConsultUserData` key-value pair of the request.

Separate Method

Assigning the value `separate` to the `ConsultUserData` key tells T-Server to store an original call's user data separately from the consultation call's user data. Consequently, the data attached to the original call is only available to the original call's parties for review and change, while the data attached to the consultation call is only available to the consultation call's parties.

Inherited Method

Using the `inherited` value for the `ConsultUserData` key tells T-Server to copy the user data from the original call to the user data structure of the consultation call when the consultation call is initiated. After the consultation call is established, its user data is stored separately from the original call's user data. Further changes to the original call's user data are not available to the consultation call's parties and vice versa.

Joint Method

Using the `joint` value for the `ConsultUserData` key tells T-Server to maintain the same user data structure for the original call and for any number of derived consultation calls. The user data structure is associated with the original call, but any of the parties of the original and consultation calls can see and make changes in the common user data. T-Server associates the user data structure with the first consultation call in a consultation call chain when the original call has ended. (A consultation call chain is created when a consulted party initiates another consultation call.) If two consultation call chains are created on different legs of the ended original call, T-Server duplicates the user data structure and associates the structures with the chains independently. Further changes of user data made by the parties of one chain will not be visible to the parties of other chains.

User Data in Multi-Site Consultation Calls

For multi-site (ISCC) consultation calls, in addition to the `ConsultUserData` key or the `consult-user-data` option, setting the `use-data-from` T-Server common option also affects user data. The `use-data-from` option specifies for a given consultation call that is routed or transferred to a remote location, what the source of its values for the `UserData` and `ConnID` attributes should be. (For the full description, see the "Multi-Site Support Section" of the "T-Server Common Configuration Options" chapter of the T-Server Deployment Guide.)

The following call flow example illustrates how setting these different values affects the availability of user data.

Example

An established call for DN A on site A has a `UserData` attribute with the key `M`.

1. DN A on site A initiates a conference with DN B on site B. This call has additional `UserData` with the key `C`.
2. DN A on site A completes the conference.

Events on DN B have the following `UserData` according to how you set options `consult-user-data` and `use-data-from`.

Resulting Keys for User Data Based on Use of Different Options

Value of **use-data-from**	**Value of consult-user-data**		
	separate	inherited	joint

Value of **use-data-from**	**Value of consult-user-data**		
`original`	M	M	M; C
`active`	C	M; C	M; C
`current` (1) [a]	C	M; C	M; C
`current` (2) [b]	M	M	M; C

a. If `UserData` is reported after the initiation of the conference, these keys result.
b. If `UserData` is reported after the completion of the conference, these keys result. The assumption for this case is that the option `merged-user-data` is set to main-only (the default value).

Extensions

An *extension* is a pointer to a data structure that takes into account switch-specific features and information that cannot be described by the other parameters in an event or a request. The extensions detailed in this section, however, pertain only to requests. They are applicable to all T-Servers and permit tuning of T-Server operations. Extensions for requests that apply only to particular T-Servers are described in the individual T-Server Deployment Guides.

> **Important**
>
> T-Library Events for information on the key-value pairs (the extensions) that appear in the `Extensions` attribute of events (as members of the TEvent structure).

Hardware Reasons in Extensions

In most cases, hardware-issued reasons are noted in the `ReasonCode` key-value pair in the `Extensions` attribute of four specific function calls and their corresponding events. (See TAgentLogin, TAgentLogout, TAgentSetReady, and TAgentSetNotReady for more information.) However, such issues are not limited to being present in those four functions (or their corresponding events).

Hardware-based reasons, as passed by extensions, are handled differently than Genesys reasons. For an illustration of the handling differences between hardware-based reasons and Genesys reasons, see the Genesys Reasons versus Media-Device Reasons diagram.

Extensions Common to All T-Servers According to Request

This section details the relationship between certain extensions and the relationships they may have with specific requests.

ISCC Extensions

> **Important**
>
> With the 7.0 release of T-Library, the `iscc` prefix for extensions does not necessarily mean that multiple sites are involved in a given request. These extensions are now supported for single-site scenarios as well.

The `Extensions` attributes in the following table apply to requests passed between T-Servers. (In multi-site environments, these are ISCC-processed requests.) The requests that use these extensions include the functions `TMakeCall()`, `TInitiateConference()`, `TInitiateTransfer()`, `TMuteTransfer()`, `TSingleStepTransfer()`, `TRouteCall()`, and `TGetAccessNumber()`.

Extensions Common to All ISCC-Processed Requests

Key	Value	Value Description
iscc-pass-extension[a]	string (`local, remote`, or `both`)	Controls where extension attributes are passed from an original client request.
iscc-xaction-type	integer (or string value of one of the enumerated route types)	Routing type to be used. See `TXRouteType()` in your API reference for the complete list of route types available.
iscc-ar-agent-dn	string	Destination DN
iscc-ar-agent-id	string	Destination agent's ID
iscc-ar-place	string	The destination agent's place
iscc-ar-duration	integer	Required time that an agent is to be reserved, in milliseconds (Specify `-1` to use the default of `100000` milliseconds.)
iscc-ar-priority	integer	Requested priority (Specify `-1` to use default of `0`, the lowest priority value.)
iscc-ar-priority-1	integer	Additional granularity for setting priority. These are only evaluated if there are several concurrent requests with the same `iscc-ar-priority`. If any of these sub-priorities is absent, its value is assumed to be `0`.
iscc-ar-priority-2	integer	
iscc-ar-priority-3	integer	

a. Exception: Do not use `iscc-pass-extension` with the `TGetAccessNumber()` function.
Note: To insure backward compatibility to 6.x and in order to comply with certain T-Servers, the default value for this key-value pair is both. An explicit value for iscc-pass-extensions is expected with ISCC requests when the same extensions interfere with origination and destination T-Servers. When the iscc-pass-extensions pair contains an invalid value, the default value is used.

For all requests taking place between T-Servers, when the `cast-type` has the value `route`, extensions are passed to the media device noted in the function `TRouteCall()` from the `ExtRoutePoint` to the requested destination. After the first unsuccessful attempt to route a call that arrives at a final destination, subsequent `TRouteCall()` requests do not contain extensions from an original client's request.

TAgentLogin, TAgentLogout, TAgentSetReady, and TAgentSetNotReady

The `Extensions` attribute listed in the following table is used to pass hardware reason codes and pertains to all of the following functions: `TAgentLogin()`, `TAgentLogout()`, `TAgentSetReady()`, and `TAgentSetNotReady()`. Recall that these are the typical, but not the exclusive, functions where hardware-reason codes are found.

Extensions in TAgentLogin, TAgentLogout, TAgentSetReady, and TAgentSetNotReady

Key	Value	Value Description
ReasonCode	String	A hardware reason code available for communication through these four function calls.

Important

The corresponding events (`EventAgentLogin`, `EventAgentLogout`, `EventAgentReady`, and `EventAgentNotReady`) for these four functions communicate the hardware reason codes as well.

TMakePredictiveCall

The `Extensions` attributes listed in the following tables (for Call Progress Detection (CPD) and for Call Party Number (CPN)) pertain to the function `TMakePredictiveCall()`, but are not applicable to all switches. Furthermore, some switches support only some subset of these extensions. Refer to individual T-Server deployment guides for applicability.

CPD-Related Extensions in TMakePredictiveCall

Key	Value	Value Description
VoiceDest	Any valid ACD Queue or Routing Point	A Queue or Routing Point to which an outbound call answered by a live voice will be transferred
AnsMachine	Any valid ACD Queue or Routing Point	A Queue or Routing Point to which an outbound call answered by an answering machine will be transferred
FaxDest	Any valid ACD Queue or Routing Point	A Queue or Routing Point to which an outbound call answered by a fax machine will be transferred

Important

Depending on switch capabilities and the means of call answer detection, T-Server can route an answered outbound call to a target DN specified in `Extensions` using the `VoiceDest, AnsMachine`, or `FaxDest` key.

CPN-Related Extensions in TMakePredictiveCall

Key	Type	Required	Default	Value Description
CPNType	KVTypeInt	No	0	Number type
CPNPlan	KVTypeInt	No	0	Numbering plan

Key	Type	Required	Default	Value Description
CPN-Presentation	KVTypeInt	No	`0`	Presentation indicator
CPN-Screening	KVTypeInt			Screening indicator
CPNDigits	KVType-String	Yes		A5 characters, according to formats specified in the appropriate numbering/dialing plan

TInitiateConference, TInitiateTransfer, and TMuteTransfer

The `Extensions` attribute listed in the following table applies to the functions `TInitiateConference()`, `TInitiateTransfer()`, and `TMuteTransfer()`. A more detailed description of `ConsultUserData` appears under User Data in Consultation Calls.

Extensions in TInitiateConference, TInitiateTransfer, TMuteTransfer

Key	Value	Value Description
ConsultUserData	default	The method specified in the `consult-user-data` configuration option is used.
	separate	User data for the consultation call is attached and stored separately from the user data attached to the original call.
	inherited	User data attached to the original call is copied to the consultation call at the moment the consultation call is initiated; after that, any changes to the original call's user data will not affect the consultation call's user data and vice versa.
	joint	User data attached to either the original or the consultation call is associated with the original call and, yet, can be seen and changed by the parties of both calls.

TReserveAgent

The `Extensions` attributes listed in the following table pertain to the function `TReserveAgent()`.

Extensions in TReserveAgent

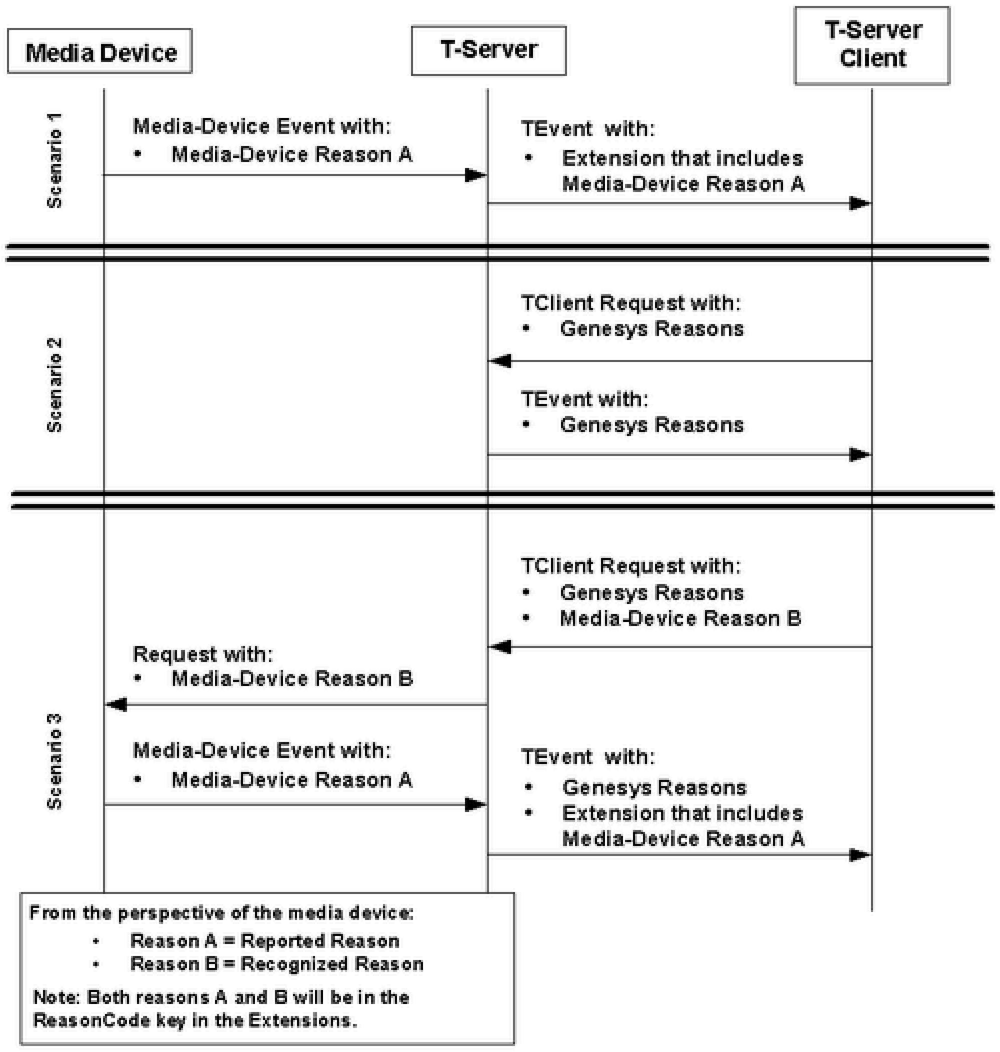
Key	Value	Value Description
ar-priority-1	integer	Additional granularity for setting priority. These are only evaluated if there are several concurrent requests with the same value for the parameter `priority`. If any of these sub-priorities is absent, its value is assumed to be `0`.
ar-priority-2	integer	
ar-priority-3	integer	

Reasons

A *reason* is a pointer to an additional data structure that provides information on causes for, and results of, actions taken by the user of `ThisDN`. If the `Reasons` attribute appears in the `TEvent` structure, it has been taken directly from the corresponding T-Library request.

Warning

There is no other source for the information found in the content of the `Reasons` attribute. Media device/hardware reason codes, and similar information, do not appear in the `Reasons` attribute. Rather, if they are supported, those hardware reasons are provided by T-Server in the `Extensions` attribute. The figure below diagrams the difference between media-device reasons and the `Reasons` attribute (Genesys Reasons).



Genesys Reasons versus Media-Device Reasons

Persistent Reasons

There are times when the value of the `Reasons` attribute does not pertain to the most recent activity related to the DN or device in question. Such reasons are considered *persistent* or *current.* In particular, `TQueryAddress` and `TRegisterAddress` return the current reason for a DN. This allows applications such as Stat Server to retrieve the state of a DN, with its associated reason, at the time of startup or re-start and improves metrics quality and accuracy.

In any given instance, the value of the `Reasons` attribute is stored by T-Server in a `TKVList`. T-Server, however, does not control the context for this list. It therefore becomes the job of the application receiving this value as part of an event to interpret it appropriately. This is not always straightforward, since T-Server only preserves one reason for a given DN at any given time.

Additionally, T-Server only stores reasons that arrive from successful requests. Thus, if you receive an error in response to, for instance, an `AgentNotReady` request (because the agent or DN could not be set to the `not ready` state), the reason that you passed with the request is not preserved, and the reason from the previous successful request is still active.

> **Important**
>
> In order to erase a reason completely, applications need to pass an empty `TKVList` in a request (and receive successful confirmation). Requests with `NULL` as the reason do not affect the current reason. (This is done to preserve backward compatibility.)

To change a reason without changing the state of the agent or DN, you can send a request with a different reason several times. (This is supported for all 7.x T-Servers, by invoking the concept of self-transition states. Some older T-Servers may return errors.)

> **Important**
>
> A T-Server synchronizes its `Reasons` attributes with its backup for use in failover. But there is no resynchronization of these attributes at the time of a backup T-Server's reconnection.

IVR Protocol Messages

This section presents detailed explanations of the messages and parameters used by the Genesys IVR XML protocol. The messages in this section are those sent by the IVR Server to the IVR.

This section contains the following subsections:

- General Messages
- Routing Messages
- Call Treatment Messages
- External Routing Messages
- Transfer/Conferencing Messages
- Call Information Messages
- Statistics Messages
- User Data Messages
- Outbound Messages

General Messages

This section includes the responses to login, logging, and reset messages.

LoginResp

This message is sent by the IVR Server to the IVR in response to a `LoginReq` message.

The `Status` parameter of the `LoginResp` message has the following possible values:

- `NoSuchClient`—There is no IVR object configured in the Configuration Layer with the name supplied in the `ClientName` parameter of the `LoginReq` message.
- `InitInProgress`—The IVR Server is in the process of initializing and is not ready to process new calls.
- `OK`—Initialization is complete, and the IVR Server can process calls.

Place required configuration information in the data transport section of the IVR `Application` object in Configuration Manager. If you do this, the information is returned in the `ConfigOptions` section of the `LoginResp` message.

LoginResp Message

Message	Parameter		Optional/ Required
Name	**Value**		
LoginResp	IServerVersion		Required
	Result	Success InvalidProtocolVersion	Required
	ConfigOptions		Optional
	Status	NoSuchClient InitInProgress OK	Optional

MonitorInfo

This message will be sent when a significant event occurs related to the server monitoring. These will be events pertinent to managing agent status. The `ReqId` parameter will be present when this event is in response to an XML request, as opposed to an unsolicited event.

MonitorInfo Message

Message	Parameter		Optional/Required
Name	**Value**		
MonitorInfo	ReqId		Optional

Server Subtype

A `Server` type of `MonitorInfo` message is created when the information being sent is related to T-Server connections. This message is never directly requested by a client, so the `ReqId` parameter of the `MonitorInfo` message will never be supplied.

This message will be sent when either an `EventLinkDisconnected` or `EventLinkConnected` event occurs, or when the T-Server socket is closed. For this event to be forwarded, it must occur on a T-Server that is used by the IVR. This is based upon the configuration of the IVR in `ConfigServer` and the name provided by the login request.

Server Status Events

T-Library Event	XML
EventLinkConnected	<Server Status='OK'/>
EventLinkDisconnected/Socket Closed/ No Connection	<Server Status='Unavailable'/>

Server Message

Message	Parameter		Optional/Required
Name	**Value**		
Server	Name		Required
	Status	OK Unavailable	Required
	Switch		Optional

Port Subtype

This message will be sent to inform the client that no further successful requests can be submitted for that port due to configuration database changes. As with the `Server` subtype; this message will never have a `ReqId` associated with it.

Port Message

Message	Parameter		Optional/Required
Name	**Value**		
Port	PortNum		Required
	Status	OK Unavailable	Required

Agent Subtype

Agent related events occurring on relevant ports are conveyed using the Agent subtype. These messages can either be in response to control messages, or due to external sources. When in response to a control message, ReqId from that related message will be used. The following table shows the relationship between T-Library events and XML messaging.

Agent Status Events

T-Library Event	XML
EventAgentLogin	<Agent PortNum='01' Status='LoggedIn'/>
EventAgentLogout	<Agent PortNum='01' Status='LoggedOut'/>
EventAgentReady	<Agent PortNum='01' Status='Ready'/>
EventAgentNotReady	<Agent PortNum='01' Status='NotReady'/>

In T-Library, the LoggedIn state is not a steady state, it only indicates that the login was successful. Another status message will always follow the LoggedIn indication to signify whether the agent is in the ready or not ready state. This is a function of the switch and may be one or the other depending on configuration. Therefore, Ready and NotReady imply LoggedIn.

It is also important to note that the query event may return an Unknown state from the switch. As a general rule, treat Unknown as LoggedOut.

Agent Message

Message	Parameter		Optional/Required
Name	**Value**		
Agent	PortNum		Required
	Status	LoggedIn LoggedOut Ready NotReady Unknown	Required

Routing Messages

This message is the basic response resulting from requests to start a call, route it, confirm the connection or indicate failure to connect, and end the call.

RouteResponse

This message is sent by the IVR Server to the IVR to indicate that the call should be routed to the specified destination.

RouteResponse Message

Message	Parameter		Optional/ Required
	Name	Value	
RouteResponse	RouteType	Default Normal Reroute RerouteAttended RerouteConferenced	Present only if supplied by URS.
	Dest		Optional
	ExtnsEx		Optional

Call Treatment Messages

Call treatment messages are used to start and control an external application that processes a call and which might return data that can then be used to route the call.

TreatCall

This message is sent by the IVR Server to the IVR to indicate that the specified call treatment should be run by the IVR.

TreatCall Message

Message	Parameter Name	Optional/Required
TreatCall	CallId	Required
	Type	Required
	Parameters	Optional
	ExtnsEx	Optional

Cancel

This message is sent by the IVR Server to the IVR to indicate that a previously started call treatment process must be canceled.

Cancel Message

Message	Parameter Name	Optional/Required
Cancel		

External Routing Messages

This message is used as a response to a request for an inter-switch transfer.

AccessNumResp

This message is sent by the IVR Server to the IVR to indicate the result of a previous `AccessNumGet/AccessNumCancel`. The `Action` parameter indicates to which type of request this message is in response. The access number is only present for a successful `AccessNumGet`.

AccessNumResp Message

Message	Parameter		Optional/Required
	Name	**Value**	
AccessNumResp	Action	Get Cancel	Required
	Result	Success Failure	Required
	AccessNum		Optional

Transfer/Conferencing Messages

These messages are used to monitor call transfers and conferencing.

CallStatus

This message is sent by the IVR Server to inform the IVR of certain call events. The list of possible events are alternatives. Only one parameter from this list appears in any message.

CallStatus Message

Message	Parameter		Optional/Required
Name	**Value**		
CallStatus	Event	Dialing Ringing Established Retrieved Busy Held ConfPartyAdd ConfPartyDel XferComplete Released	Required

CallError

This message is sent by the IVR Server to inform the IVR that an error occurred during the setup of a transfer or a conference call.

Errors related to agent control activities will be represented by the `AgentControl` or the `NotAllowed` indication. When the error is due to an `EventError`, the `TlibErrCode` will be populated with `AttributeErrorCode` and the type will be `AgentControl`. `NotAllowed` will be used exclusively when attempting to control a server controlled port. The user supplied `ReqId` will be returned in the error.

CallError Message

Message	Parameter		Optional/Required
Name	**Value**		
CallError	FailedReq	Unknown NoSuchCall OneStepXfer OneStepConf	Required

Message	Parameter		Optional/Required
		InitConf CompleteConf InitXfer CompleteXfer RetrieveCall MakeCall AgentControl NotAllowed	
	TLibErrCode		Optional
	ReqId		Optional

Call Information Messages

This message is in response to a request for data attached to the call.

CallInfoResp

The response contains information on all of the listed parameters for which data has been collected.

> **Important**
> The value of the `FirstHomeLocation` parameter is only returned for logins with version 3.0 or later of the `IServer.dtd` file. This attribute corresponds to T-Library's `AttributeFirstTransferHomeLocation` attribute. See T-Library Events for details.

CallInfoResp Message

Message	Parameter Name	Optional/Required
CallInfoResp	ANI	Optional
	DNIS	Optional
	CalledNum	Optional
	ConnId	Optional
	TSCallId	Optional
	PortDN	Optional
	PortTrunk	Optional
	PortQueue	Optional
	OtherDN	Optional
	OtherTrunk	Optional
	OtherQueue	Optional
	LastEvent (The most recently recorded T-Server event.)	Optional
	FirstHomeLocation	Optional

Statistics Messages

The statistics message enables you to receive data on the `CurrNumberWaitingCalls` and `ExpectedWaitTime` statistics. These statistics must be configured in Stat Server before they can be accessed through the IVR Server.

StatResp

Supplies the response to requests for statistics.

StatResp Message

Message	Parameter		Optional/Required
	Name	**Value**	
StatResp	RequestId		Required
	ResultCode	Success NoSuchStat MiscError	Required
	Result		Optional

User Data Messages

This message is in response to a request to access and control data about the actions performed by callers.

UDataResp

This message contains the response to the previous user data messages. The responses for `UDataSet` and `UDataDel` indicate either success or, if failure, the reason for the failure.

The response for a successful `UDataGet` includes the values for the requested keys.

UDataResp Message

Message	Parameter		Optional/ Required
Name	**Value**		
UDataResp	RequestId		Required
	Result	Success NoSuchCall NoMatch FeatureNotSupported MiscError	Required
	UDataEx		Optional

Outbound Messages

DialOutRegistryResp

Sent from IVR Server to the IVR, this message returns information about the related `DialOutRegistry` message. `ConfigError` is returned when the corresponding DN from the `DialOutRegistry` message either is not defined in the Configuration Layer or is not a route point. `MiscFailure` is currently not used. `Success` will be returned in all other cases. When using commands `Remove` and `RemoveAll`, `Success` will always be returned.

DialOutRegistryResp Message

Message	Parameter		Optional/Required
Name	**Value**		
DialOutRegistryResp	Result	MiscFailure ConfigError Success	Required

DialOut

Sent from IVR Server to the IVR, this message indicates that an outbound call has been requested. Values from the original `TMakePredictiveCall` are included in this message where `UDataEx` and `ExtnsEx` are `AttributeUserData` and `AttributeExtensions`, respectively. Also, `OrigNum` is retrieved from `AttributeThisDN` and `DestNum` is `AttributeOtherDN`. `TimeToAnswer` gives the amount of time, in seconds, that the IVR should allow for an outbound call to be answered before a `NoAnswer` failure should be returned.

DialOut Message

Message	Parameter		Optional/Required
Name	**Value**		
DialOut	OrigNum		Required
	DestNum		Required
	RefID		Required
	TimeToAnswer		Required

Genesys Interaction Models

Genesys Interaction Models section of this document contains information on a selected list of call and interaction models. This information ranges from basic call scenarios to complex, but common ones. Based on the history of how this information has been presented in the past in various documents, model details may differ from section to section. This information appears in the following sections:

- Call Models and Flows presents the bulk of Genesys voice-based models. In this case, you can view call model details, and network attended transfer call flows. In each case, selected event information is provided to enhance your understanding of the model.
- Basic Interaction Models offers a look at selected models for interactions of non-voice type.
- IVR Call Flows contains the standard IVR call flows, with annotations.

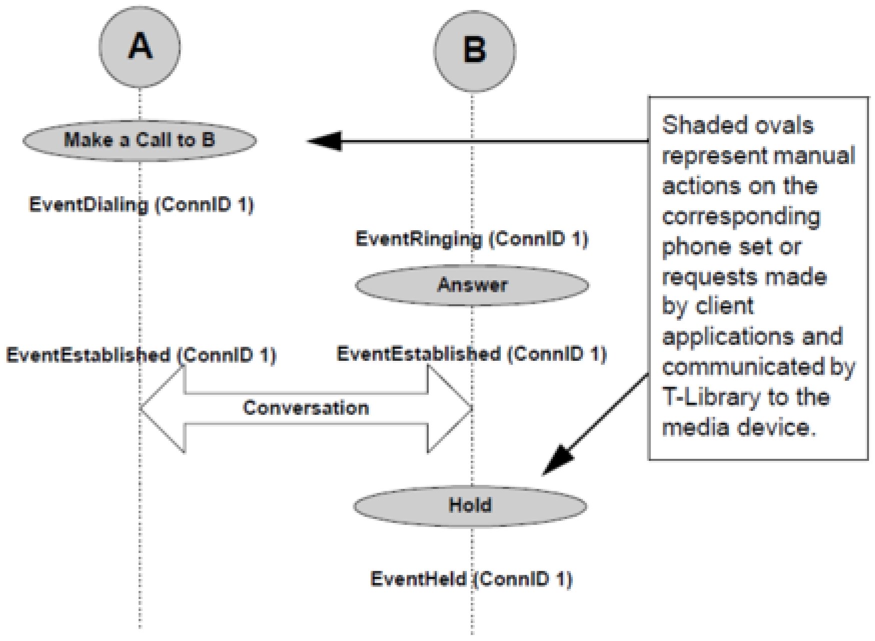
Call Models and Flows

Legend

All parties shown in a call scenario, except where stated explicitly, are considered internal and are monitored by T-Server. If one or more external parties participated in the call, the following apply:

- T-Server will not distribute any events to the external (nonmonitored) party.
- T-Server may not have any information about the nonmonitored party, so its reference may not be specified.

The following diagram illustrates a basic call model.


A
B
Make a Call to B
EventDialing (ConnID 1)
EventRinging (ConnID 1)
Answer
EventEstablished (ConnID 1)
EventEstablished (ConnID 1)
Conversation
Hold
EventHeld (ConnID 1)
Shaded ovals represent manual actions on the corresponding phone set or requests made by client applications and communicated by T-Library to the media device.

Sample Call Model

Activities like conference and transfer can be performed to an existing multi-party call (a conference call). When so, Party A is considered a "complex party" and the following apply:

- Events assigned to Party A, as shown in call scenarios, are sent to every party of "complex party."
- Reference to Party A in `AttributeOtherDN` are not present.

This is also represented in the following tables.

Depiction of Complex Parties in Call Models 1

PARTY A (complex)	PARTY C
EventPartyChanged ConnID **1** ThisDN **A** OtherDN **C**	**EventPartyChanged** ConnID **1** ThisDN **C** OtherDN **A**

Depiction of Complex Parties in Call Models 2

PARTY A0	PARTY A1	PARTY C
EventPartyChanged ConnID **1** ThisDN **A0** OtherDN **C**	**EventPartyChanged** ConnID **1** ThisDN **A1** OtherDN **C**	**EventPartyChanged** ConnID **1** ThisDN **C**

Since T-Library is a superset of functions, not every scenario described in this document is supported by every type of switch. For more details, see the T-Server Deployment Guide that applies to your T-Server/switch pair.

When more than one event is presented in one table cell, the order in which the events are distributed may vary.

Attributes `ThirdPartyDN` and `ThirdPartyDN` Role specify DNs to which two-step operations are initiated and completed.

A call is considered to be queued until either `EventDiverted` or `EventAbandoned` regarding the queue is generated.

Comments

Note the following comments in the call models:

`*OPT`—Optional.

`*DIAL`—May be a dialed number or is not present if T-Server has no information about the other party.

List of Call Models

Basic Call Models

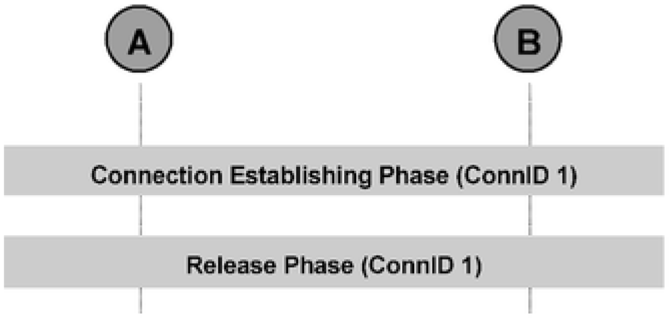
This section documents the basic scenarios under which calls arrive in a contact center.

Note the following comments in the call models:

*OPT—Optional.

*DIAL—May be a dialed number or is not present if T-Server has no information about the other party.

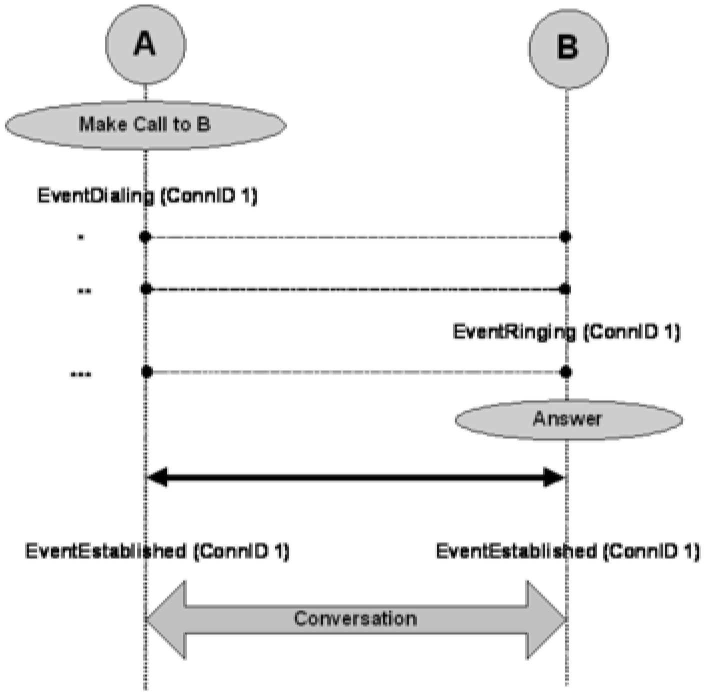
Simple Call Model



Simple Call Model

Connection-Establishing Phase (Internal/Inbound Call)

The following graphic and table describe the connection establishing phase (internal/inbound call).



Connection-Establishing Phase (Internal/Inbound Call)

PARTY A	PARTY B
Make Call to B (TMakeCall)	
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN *DIAL OtherDNRole **Destination** *DIAL	
	EventRinging ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**
	Answer (TAnswerCall)
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** OtherDNRole **Destination**	**EventEstablished** ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**

PARTY A	PARTY B
Conversation	

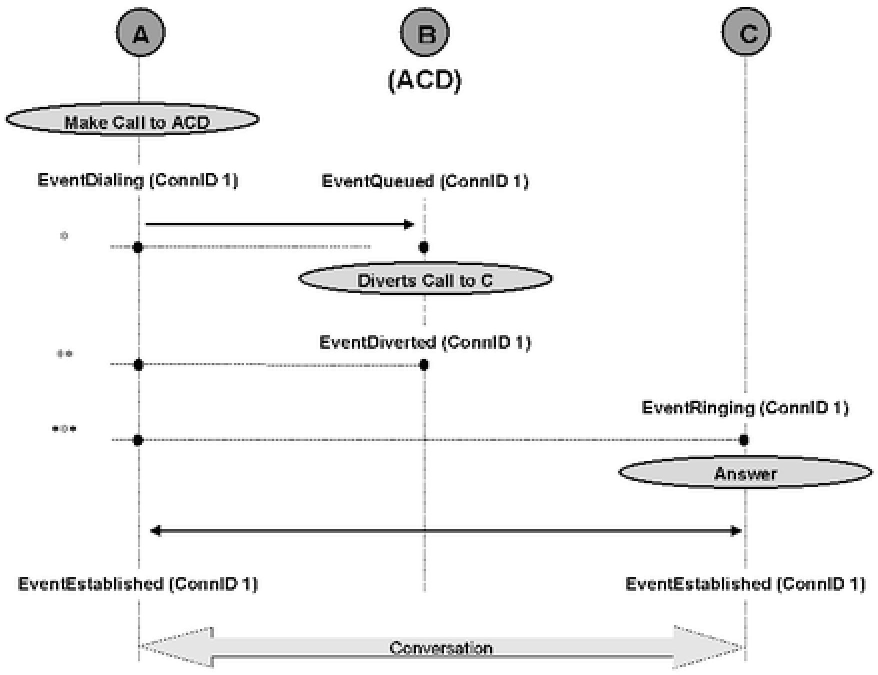
Abnormal Call Flow

Interruption Point	PARTY A	PARTY B
*	**EventReleased** ConnID **1** ThisDN **A** ThisDNRole **Origination** CallState **OK**	
**	**EventDestinationBusy** ConnID **1** ThisDN **A** ThisDNRole **Origination** CallState [a]	
***	**EventReleased** ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**

a. `CallState` may have values that clarify the reason for the destination being busy, for instance `CallState SitInvalidNum`.

Connection-Establishing Phase (Internal/Inbound Call to ACD)

The following graphic and table describe the connection establishing phase (internal/inbound call to ACD).



Connection-Establishing Phase (Internal/Inbound Call to ACD)

PARTY A	PARTY B (ACD Group)	PARTY C
Make Call to B		
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	**EventQueued** ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**	
	Diverts call to C	
	EventDiverted ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** ThirdPartyDN **C** *OPT ThirdPartyDNRole **Destination** *OPT	
		EventRinging ConnID **1** ThisDN **C** ThisQueue **B** ThisDNRole **Destination** OtherDN **A**

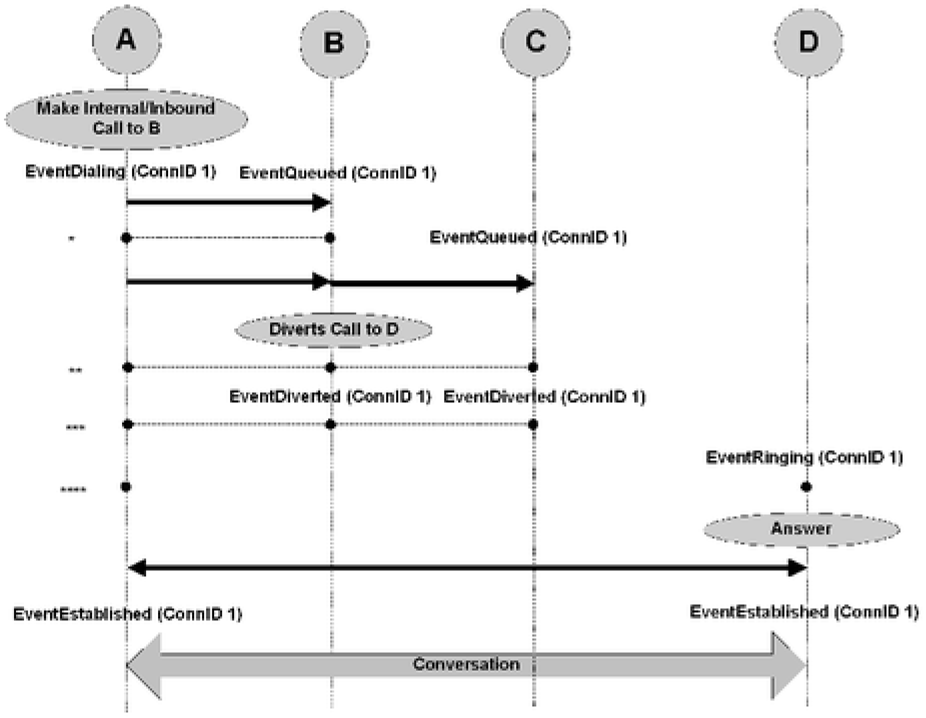
PARTY A	PARTY B (ACD Group)	PARTY C
		OtherDNRole **Origination** CallState **OK**
		Answer (TAnswerCall)
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination**		**EventEstablished** ConnID **1** ThisDN **C** ThisQueue **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**
Conversation		

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**		
***	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**

Connection-Establishing Phase (Internal/Inbound Call Queued to Multiple ACDs)

The following graphic and table describe the connection establishing phase (internal/inbound call queued to multiple ACDs).



Connection-Establishing Phase (Internal/Inbound Call Queued to Multiple ACDs)

PARTY A	PARTY B (ACD)	PARTY C (ACD)	PARTY D
Make Internal/ Inbound Call to B (ACD)			
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	**EventQueued** ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**		
		EventQueued ConnID **1** ThisDN **C** ThisQueue **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**	
	Diverts Call to D		
	EventDiverted	**EventDiverted**	

PARTY A	PARTY B (ACD)	PARTY C (ACD)	PARTY D
	ConnID **1** ThisDN **B** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination**	ConnID **1** ThisDN **C** ThisQueue **C** ThirdPartyDN **D** ThirdPartyQueue **B** CallState **Redirected** [a]	
			EventRinging ConnID **1** ThisDN **D** ThisQueue **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**
			Answer (TAnswerCall)
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **D** OtherDNRole **Destination** CallState **OK**			**EventEstablished** ConnID **1** ThisDN **D** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**
Conversation			

a. For ACD configurations where calls are distributed to agents assigned directly to ACD groups, `CallState` and its value of `Redirected` are present. For ACD configurations where calls are distributed to agents assigned to secondary ACD groups associated with top-level ACD queues, the `CallState`, with the value `Redirected`, is not present.

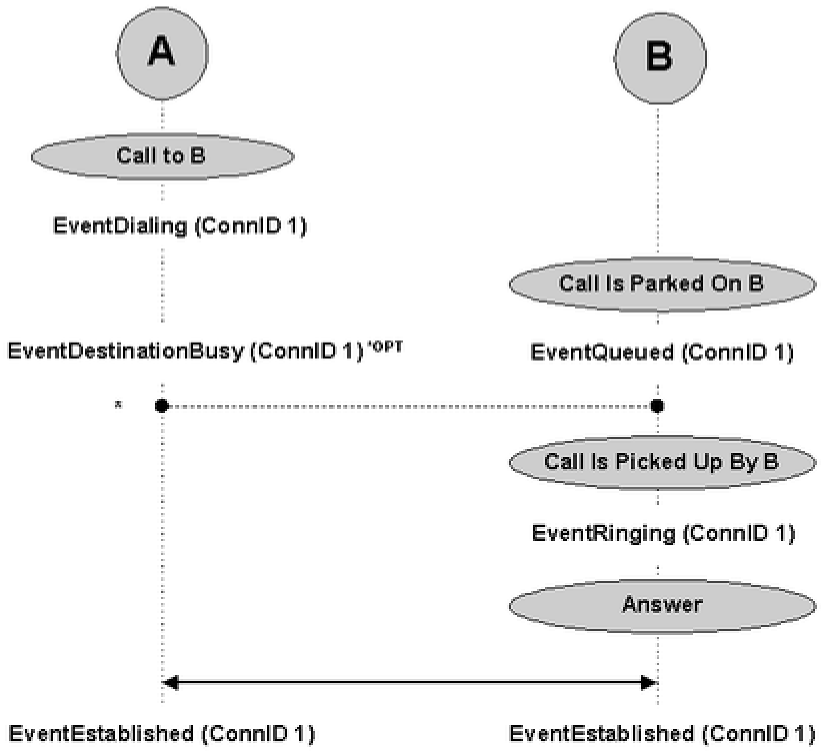
Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C	PARTY D
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** ThisQueue **B** OtherDN **A** CallState **OK**		
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** ThisQueue **B** OtherDN **A** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **C** ThisQueue **C** OtherDN **A** CallState **OK**	
***	**EventReleased**			

Interruption Point	PARTY A	PARTY B	PARTY C	PARTY D
	ConnID **1** ThisDN **A** OtherDN **D** CallState **OK**			
****	**EventReleased** ConnID **1** ThisDN **A** OtherDN **D** CallState **OK**			**EventAbandoned** ConnID **1** ThisDN **D** ThisQueue **C** OtherDN **A** CallState **OK**

Connection-Establishing Phase (Internal/Inbound Call with Call Parking)

The following graphic and table describe the connection establishing phase (internal/inbound call with call parking).



Connection-Establishing Phase (Internal/Inbound Call with Call Parking)

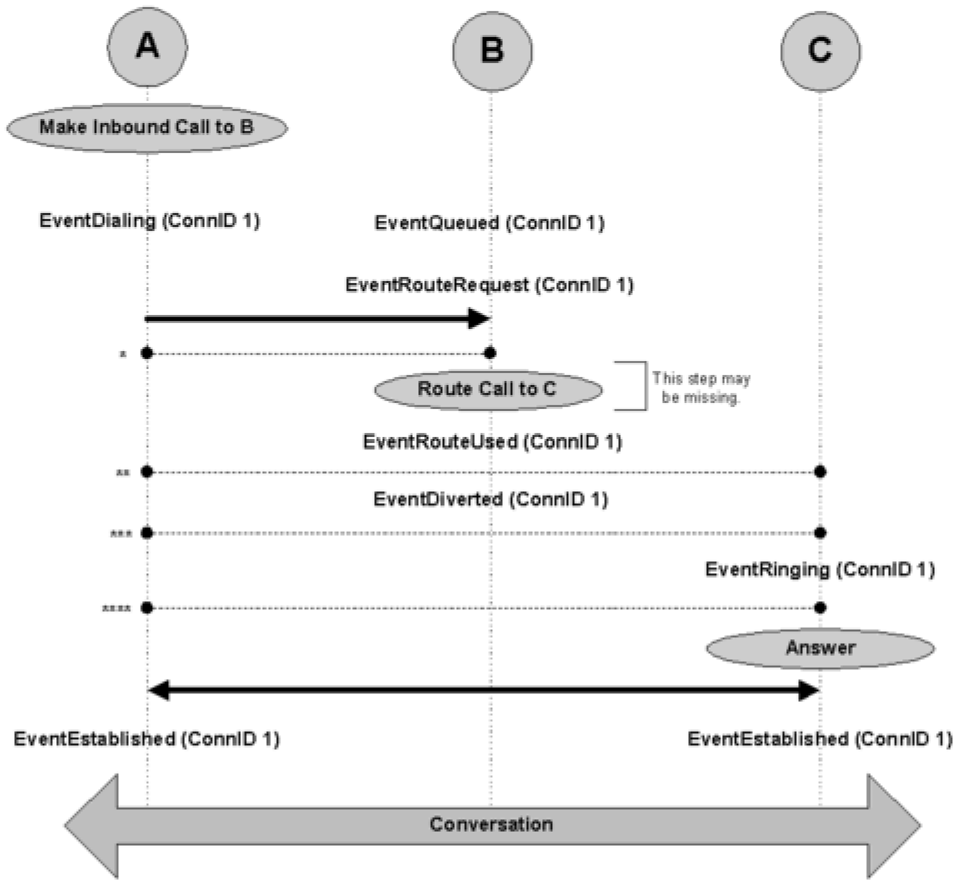
PARTY A	PARTY B
Make Call To B (TMakeCall)	
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	
	Call Is Parked On B
EventDestinationBusy *OPT ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	**EventQueued** ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**
	Call Is Picked Up By B
	EventRinging ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**
	Answer (TAnswerCall)
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** OtherDNRole **Destination**	**EventEstablished** ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**
Conversation	

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B
*	**EventReleased** ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**

Connection-Establishing Phase (Internal/Inbound Call with Routing—RouteQueue Case)

The following graphic and table describe the connection establishing phase (internal/inbound call with routing - RouteQueue case).



Connection-Establishing Phase (Internal/Inbound Call with Routing—RouteQueue Case)

PARTY A	PARTY B (Routing Point/CDN)	PARTY C
Make Incoming Call to Information Service		
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** OtherDNRole **Destination**	**EventQueued** ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** **EventRouteRequest** ConnID **1** ThisDN **B**	

PARTY A	PARTY B (Routing Point/CDN)	PARTY C
	ThisQueue **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**	
	Route Call to C [a] (TRouteCall)	
	EventRouteUsed ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** ThirdPartyDN **C** *OPT ThirdPartyDNRole **Destination** *OPT **EventDiverted** ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** ThirdPartyDN **C** *OPT ThirdPartyDNRole **Destination** *OPT	
		EventRinging ConnID **1** ThisDN **C** ThisQueue **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**
		Answer (TAnswerCall)
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination**		**EventEstablished** ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**
	Conversation	

a. `RouteCall` to C (`TRouteCall()`) may be missing.

Abnormal Call Flow

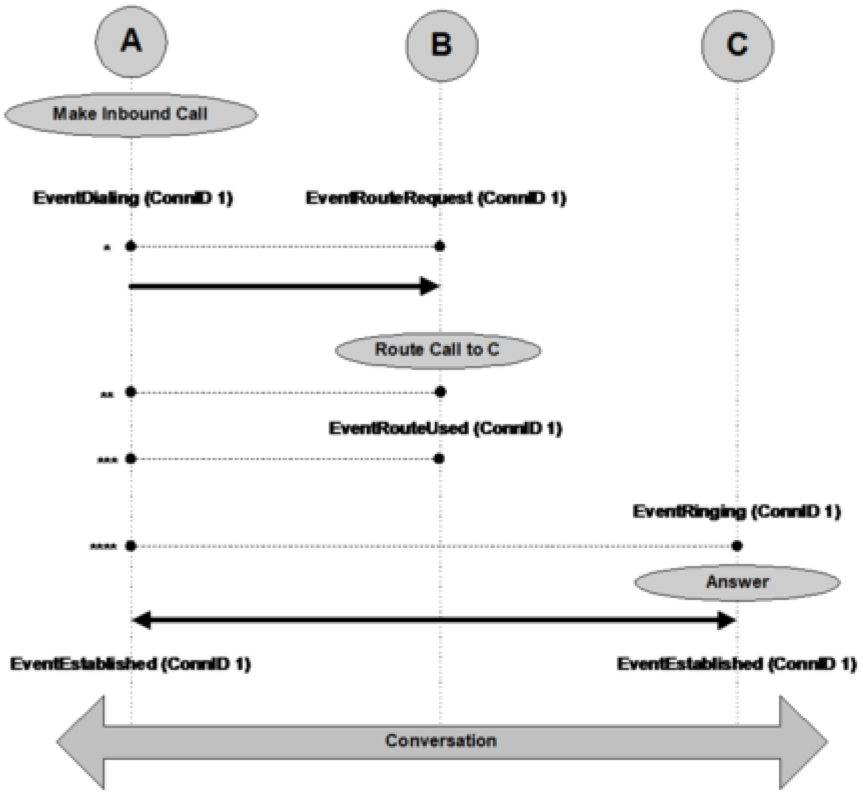
Interruption Point	PARTY A	PARTY B	PARTY C
* And	**EventReleased** ConnID **1**	**EventAbandoned** ConnID **1**	

Interruption Point	PARTY A	PARTY B	PARTY C
**	ThisDN **A** OtherDN **B** CallState **OK**	ThisDN **B** OtherDN **A** CallState **OK**	
***	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		
****	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**

a. `RouteCall` to C (`TRouteCall()`) may be missing.

Connection-Establishing Phase (Internal/Inbound Call with Routing)

The following graphic and table describe the connection establishing phase (internal/inbound call with routing).



Connection-Establishing Phase (Internal/Inbound Call with Routing)

PARTY A	PARTY B (Routing Point/CDN)	PARTY C
Make Incoming Call to Information Service		
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	**EventRouteRequest** ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**	
	Route Call to C [a] (TRouteCall)	
	EventRouteUsed ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** ThirdPartyDN **C** [b] ThirdPartyDNRole **Destination** *OPT CallState **OK/Redirected** [c]	
		EventRinging ConnID **1**

PARTY A	PARTY B (Routing Point/CDN)	PARTY C
		ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**
		Answer (TAnswerCall)
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination**		**EventEstablished** ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**
	Conversation	

a. Not present if a call has been routed by default; that is, a switch did not receive any routing instruction from a computer domain within a timeout configured on the switch side (scripted or otherwise) and therefore processed the call using switch logic. b. Content of `ThirdPartyDN` depends on the call scenario:

- If information about the destination is available at the moment `EventRouteUsed` is generated, this attribute is mandatory; a DN where the call has been delivered must be reported.
- If the information is not available, but the call has been routed through T-Server, this attribute is mandatory; a DN where the call has been sent must be reported.
- If a call has been routed to a default destination or routed by another application, this attribute is optional (depends on switch capabilities).

c. `CallState` has a value of `Redirected (22)` if a call has been routed by a switch. For Aspect ACD, Rockwell Spectrum, and Hicom 300 E CS switches, the attribute `Callstate` is not present.

Abnormal Call Flow

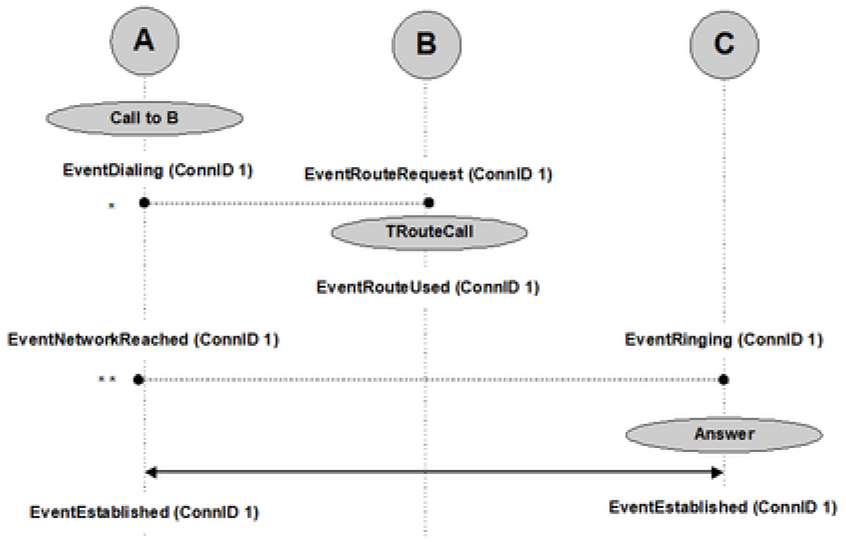
Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**	**EventAbandoned** [a] ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
***	**EventReleased**		

Interruption Point	PARTY A	PARTY B	PARTY C
	ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		
****	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**

a. `EventError` must be sent after `EventAbandoned` in this case to make the `ReferenceID` available.

Connection-Establishing Phase (Internal/Inbound Call with Routing Outbound)

The following graphic and table describe the connection establishing phase (internal/inbound call with routing outbound).


A
B
C
Call to B
EventDialing (ConnID 1)
EventRouteRequest (ConnID 1)
TRouteCall
EventRouteUsed (ConnID 1)
EventNetworkReached (ConnID 1)
EventRinging (ConnID 1)
Answer
EventEstablished (ConnID 1)
EventEstablished (ConnID 1)
Connection-Establishing Phase (Internal/Inbound Call with Routing Outbound)

PARTY A	PARTY B (Routing Point)	PARTY C
Incoming Call		
EventDialing	**EventRouteRequest**	

PARTY A	PARTY B (Routing Point)	PARTY C
ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**	
	Route Call to C [a] (TRouteCall)	
EventNetworkReached ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** *DIAL OtherDNRole **Destination** *DIAL	**EventRouteUsed** ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** ThirdPartyDN **C** [b] ThirdPartyDNRole **Destination** *OPT CallState **OK/Redirected** [c]	**EventRinging** ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**
		Answer (TAnswerCall)
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination**		**EventEstablished** ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**
Conversation		

a. Not present if a call has been routed by default; that is, a switch did not receive any routing instruction from a computer domain within a timeout configured on the switch side (scripted or otherwise) and therefore processed the call using switch logic. b. Content of `ThirdPartyDN` depends on the call scenario:

- If information about the destination is available at the moment `EventRouteUsed` is generated, this attribute is mandatory; a DN where the call has been delivered must be reported.
- If the information is not available, but the call has been routed through T-Server, this attribute is mandatory; a DN where the call has been sent must be reported.
- If a call has been routed to a default destination or routed by another application, this attribute is optional (depends on switch capabilities).

c. `CallState` has a value of `Redirected (22)` if a call has been routed by a switch. For Nortel Communication Server 1000 with SCCS MLS, Aspect ACD, Rockwell Spectrum, and Hicom 300 E CS switches, the attribute `CallState` is not present.

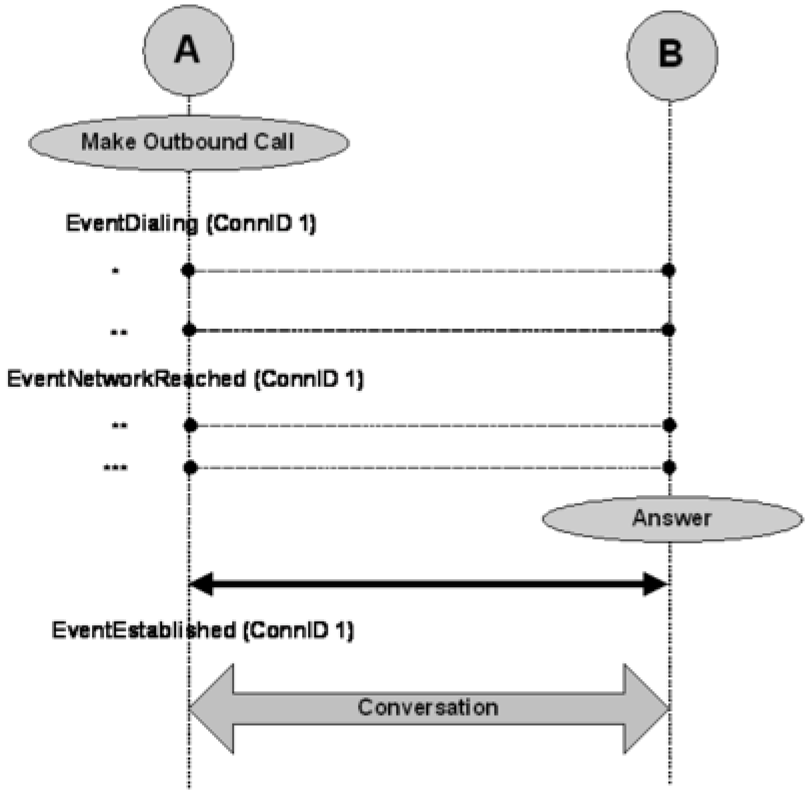
Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased**	**EventAbandoned**	

Interruption Point	PARTY A	PARTY B	PARTY C
	ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**

Connection-Establishing Phase (Outbound Call)

The following graphic and table describe the connection establishing phase (outbound call).



Connection-Establishing Phase (Outbound Call)

PARTY A	PARTY B
Make Outside Call (TMakeCall)	

PARTY A	PARTY B
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	
EventNetworkReached [a] ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	
	Answer
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *OPT OtherDNRole **Destination** *OPT	
Conversation	

a. When a switch does not report network reached, T-Server simulates `EventNetworkReached` right before distributing `EventEstablished`.

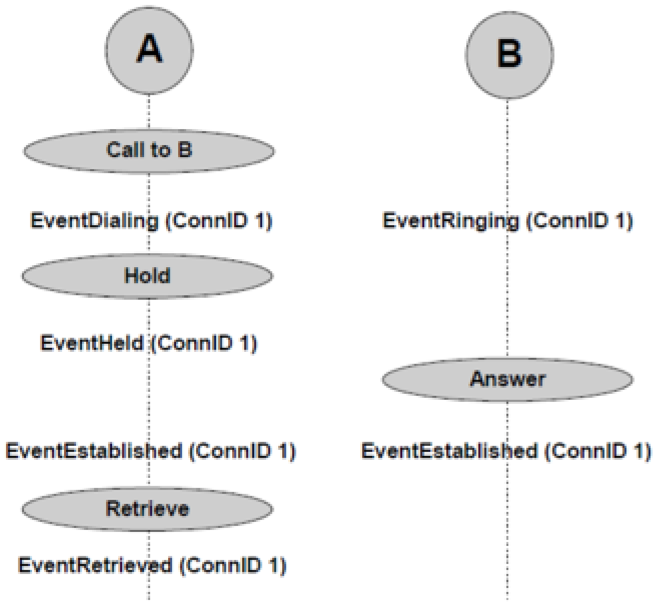
Abnormal Call Flow

Interruption Point	PARTY A
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**
**	**EventDestinationBusy** ConnID **1** ThisDN **A** OtherDN **B** CallState [a]
***	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**

a. `CallState` may have values that clarify the reason for the destination being busy, for instance `CallStateSitInvalidNum`.

Connection-Establishing Phase While On Hold (Internal/Outbound Call)

The following graphic and table describe the connection establishing phase (internal/outbound call).



Connection-Establishing Phase While On Hold (Internal/Outbound Call)

PARTY A	PARTY B
Call to B	
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** OtherDNRole **Destination** CallState **OK**	**EventRinging** ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**
Hold	
EventHeld ConnID **1**	

PARTY A	PARTY B
ThisDN **A** OtherDN **B**	
	Answer
EventEstablished ConnID **1** ThisDN **A** OtherDN **B**	**EventEstablished** ConnID **1** ThisDN **B** OtherDN **A**
Retrieve	
EventRetrieved ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	

Releasing Calls

This section illustrates the standard processes by which calls are released.

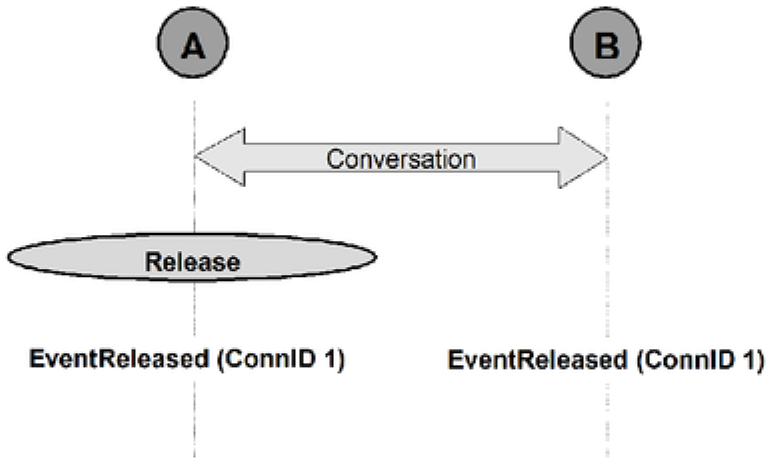
Note the following comments in the call models:

*OPT—Optional.

*DIAL—May be a dialed number or is not present if T-Server has no information about the other party.

Release Phase

The following graphic and table describe the release phase.

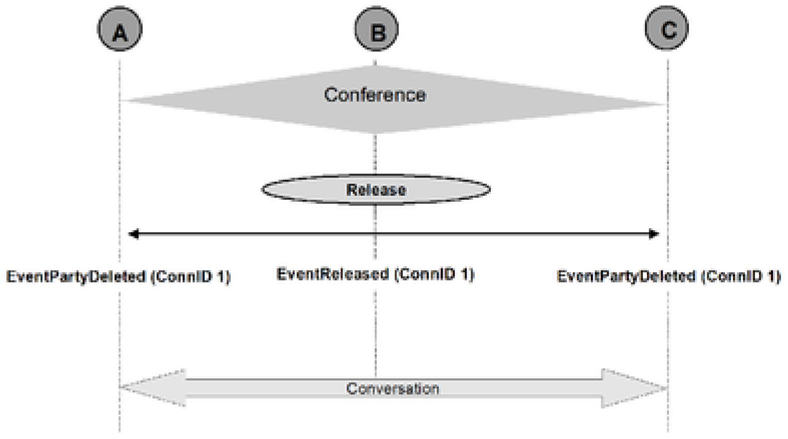


Release Phase

PARTY A	PARTY B
Conversation	
Release (TReleaseCall)	
EventReleased ConnID **1** ThisDN **A** OtherDN **B** *OPT CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** *OPT CallState **OK**

Release from Conference Phase

The following graphic and table describe the release from conference phase.



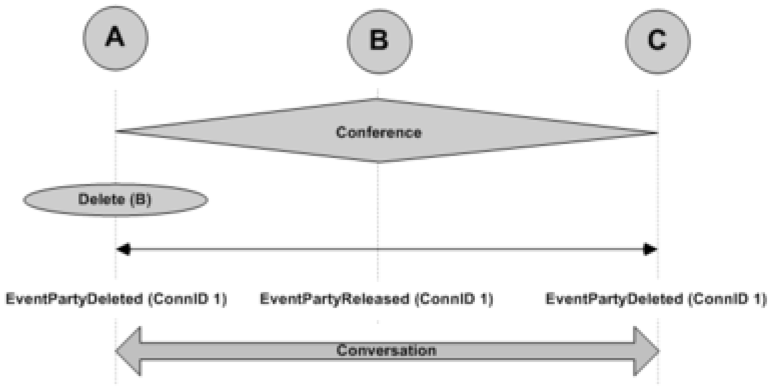
Release from Conference Phase

PARTY A	PARTY B	PARTY C
	Conference	
	Release (TReleaseCall)	
EventPartyDeleted ConnID **1** ThisDN **A** OtherDN **B** OtherDNRole **DeletedParty** ThirdPartyDN **B** ThirdPartyDNRole **DeletedBy** CallState **OK/Conferenced** [a]	**EventReleased** ConnID **1** ThisDN **B** CallState **OK**	**EventPartyDeleted** ConnID **1** ThisDN **C** OtherDN **B** OtherDNRole **DeletedParty** ThirdPartyDN **B** ThirdPartyDNRole **DeletedBy** CallState **OK/Conferenced** [a]
Conversation		

a. If more than two parties remain in the conference call, `CallState` has a value of `Conferenced`; otherwise, `CallState` has a value of `OK`.

Delete from Conference Phase

The following graphic and table describe the release phase.


A
B
C
Conference
Delete (B)
EventPartyDeleted (ConnID 1)
EventPartyReleased (ConnID 1)
EventPartyDeleted (ConnID 1)
Conversation

Delete from Conference Phase

PARTY A	PARTY B	PARTY C
	Conference	
Delete B (TDeleteFromConference)		
EventPartyDeleted ConnID **1** ThisDN **A** OtherDN **B** OtherDNRole **DeletedParty** ThirdPartyDN **A** ThirdPartyDNRole **DeletedBy** CallState **OK/Conferenced** [a]	**EventReleased** ConnID **1** ThisDN **B** CallState **OK**	**EventPartyDeleted** ConnID **1** ThisDN **C** OtherDN **B** OtherDNRole **DeletedParty** ThirdPartyDN **A** ThirdPartyDNRole **DeletedBy** CallState **OK/Conferenced** [a]
	Conversation	

a. If more than two parties remain in the conference call, `CallState` has a value of `Conferenced`; otherwise, `CallState` has a value of `OK`.

Holding, Transferring, and Conferencing

The call models here show the functions and events related to placing calls on hold, transferring calls, and creating conference calls.

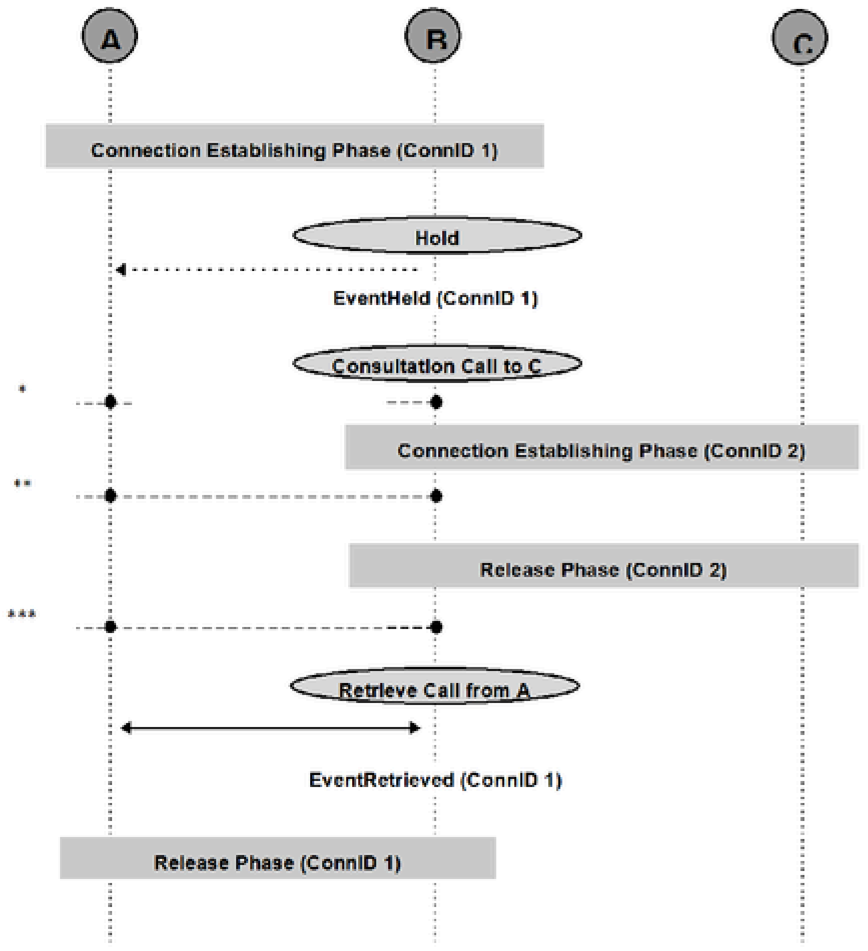
Note the following comments in the call models:

`*OPT`—Optional.

`*DIAL`—May be a dialed number or is not present if T-Server has no information about the other party.

Hold/Retrieve Function, Consulted Party Answers

The following graphic and table describe the hold/retrieve function, when the consulted party answers.



Hold/Retrieve Function, Consulted Party Answers

PARTY A	PARTY B	PARTY C
	Call-Establishing Phase (ConnID 1)	
	Hold (THoldCall)	
	EventHeld ConnID **1** ThisDN **B** OtherDN **A**	
	Make Call to C (Consultation) (TMakeCall)	
	Call-Establishing Phase (ConnID 2)	

PARTY A	PARTY B	PARTY C
Release Phase (ConnID 2)		
	Retrieve Call from A (TRetrieveCall)	
	EventRetrieved [a] ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
Release Phase (ConnID 1)		

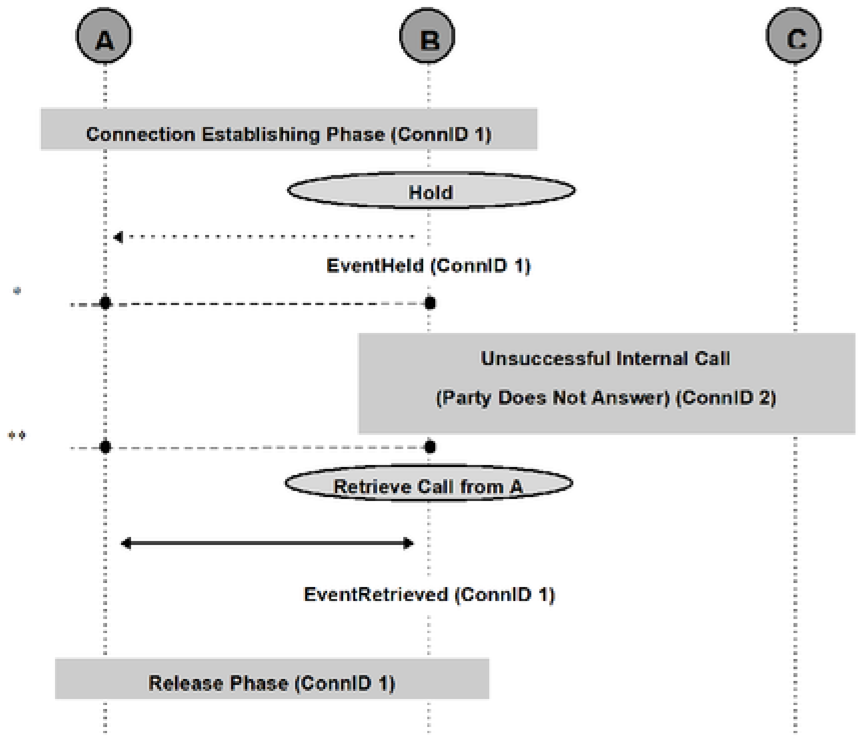
a. With `EventRetrieved`, the values for attributes `ThisDNRole` and `ThisQueue` are the same as those for the attributes of the same names, if any, in the events preceding `EventRetrieved` (`EventEstablished` and `EvenRinging`). For non-ACD calls, however, `ThisQueue` is not reported.

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**
***	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**

Hold/Retrieve Function, Consulted Party Does Not Answer

The following graphic and table describe the hold/retrieve function, when the consulted party does not answer.


A
B
C
Connection Establishing Phase (ConnID 1)
Hold
EventHeld (ConnID 1)
*
Unsuccessful Internal Call
(Party Does Not Answer) (ConnID 2)
**
Retrieve Call from A
EventRetrieved (ConnID 1)
Release Phase (ConnID 1)

Hold/Retrieve Function, Consulted Party Does Not Answer

PARTY A	PARTY B	PARTY C
	Call-Establishing Phase (ConnID 1)	
	Hold (THoldCall)	
	EventHeld ConnID **1** ThisDN **B** OtherDN **A**	
	Unsuccessful Internal Call (Party Does Not Answer) (ConnID 2)	
	Retrieve Call from A (TRetrieveCall)	
	EventRetrieved [a] ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
	Release Phase (ConnID 1)	

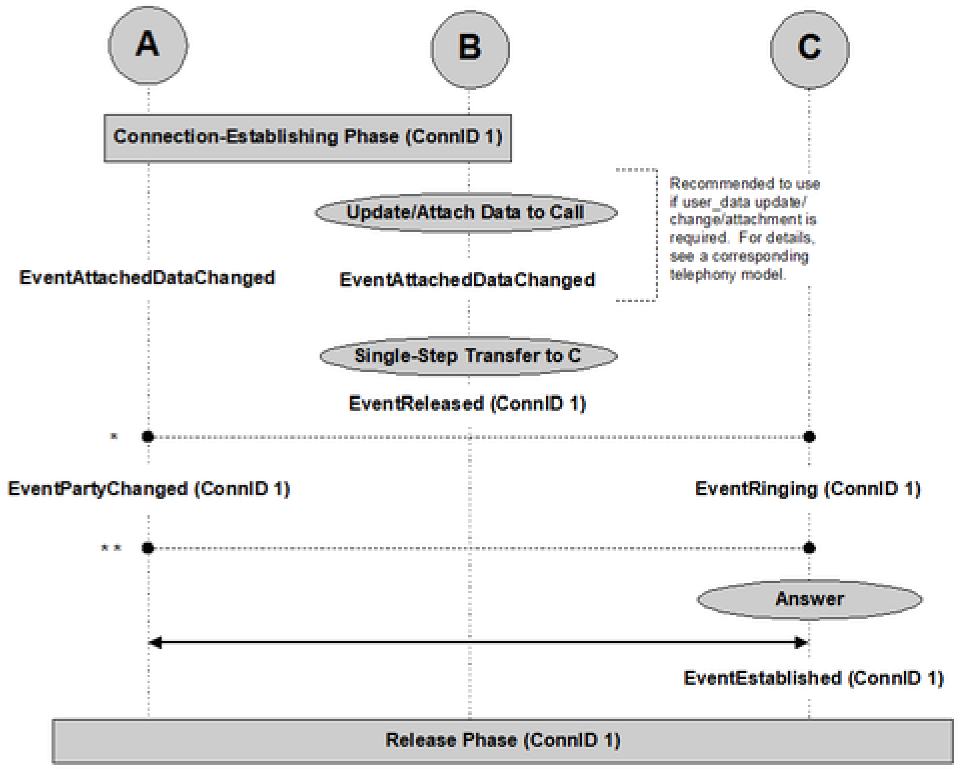
a. With EventRetrieved, the values for attributes ThisDNRole and ThisQueue are the same as those for the attributes of the same names, if any, in the events preceding EventRetrieved (EventEstablished and EvenRinging). For non-ACD calls, however, ThisQueue is not reported.

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**

Single-Step Transfer

The following graphic and table describe a single-step transfer.


A
B
C
Connection-Establishing Phase (ConnID 1)
Update/Attach Data to Call
Recommended to use if user_data update/ change/attachment is required. For details, see a corresponding telephony model.
EventAttachedDataChanged
EventAttachedDataChanged
Single-Step Transfer to C
EventReleased (ConnID 1)
*
EventPartyChanged (ConnID 1)
EventRinging (ConnID 1)
* *
Answer
EventEstablished (ConnID 1)
Release Phase (ConnID 1)

Single-Step Transfer

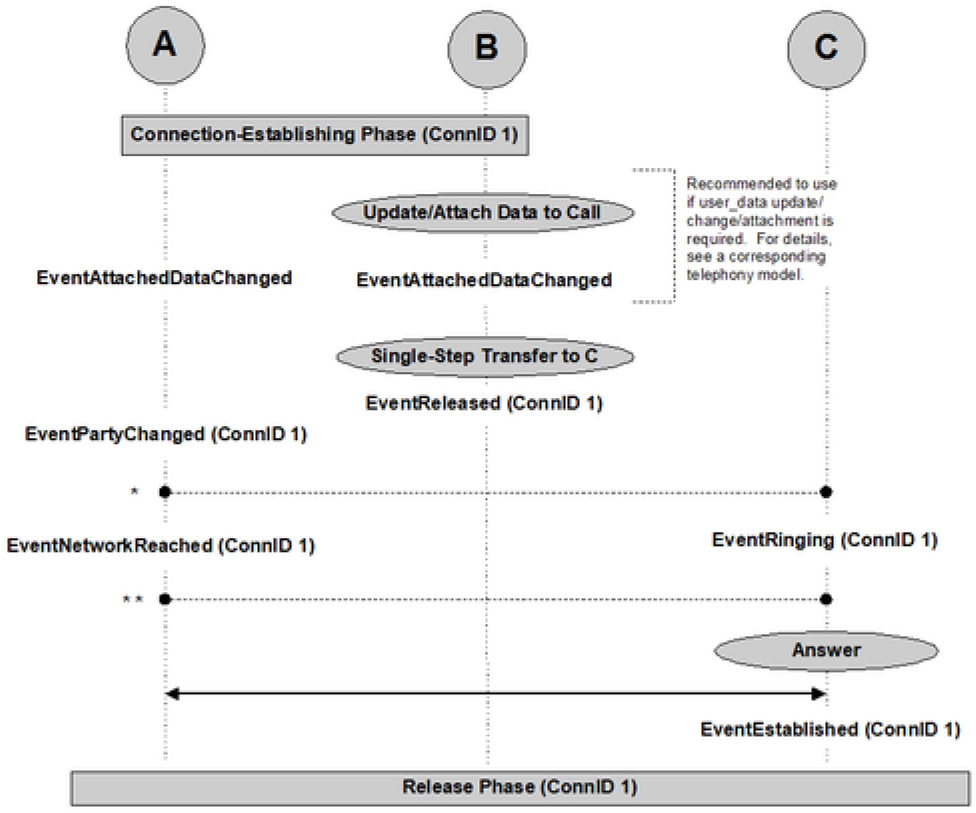
PARTY A	PARTY B	PARTY C
	Call-Establishing Phase (ConnID 1)	
	Single-Step Transfer to C (TSingleStepTransfer)	
EventPartyChanged ConnID **1** PreviousConnID **1** ThisDN **A** OtherDN **C** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred**	**EventReleased** ConnID **1** ThisDN **B** ThirdPartyDN **C** OtherDN **A** CallState **Transferred** Cause **1stepTransfer**	**EventRinging** ConnID **1** ThisDN **C** OtherDN **A** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred**
		Answer (TAnswerCall)
		EventEstablished ConnID **1** ThisDN **C** OtherDN **A**

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**

Single-Step Transfer (Outbound)

The following graphic and table describe a single-step transfer (outbound).


A
B
C
Connection-Establishing Phase (ConnID 1)
Update/Attach Data to Call
Recommended to use if user_data update/ change/attachment is required. For details, see a corresponding telephony model.
EventAttachedDataChanged
EventAttachedDataChanged
Single-Step Transfer to C
EventReleased (ConnID 1)
EventPartyChanged (ConnID 1)
*
EventNetworkReached (ConnID 1)
EventRinging (ConnID 1)
**
Answer
EventEstablished (ConnID 1)
Release Phase (ConnID 1)

Single-Step Transfer (Outbound)

PARTY A	PARTY B	PARTY C
	Call-Establishing Phase (ConnID 1)	
	Single-Step Transfer to C (TSingleStepTransfer)	
EventPartyChanged ConnID **1** PreviousConnID **1** ThisDN **A** OtherDN **C** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred** **EventNetworkReached** ConnID **1** ThisDN **A** OtherDN **C** *DIAL OtherDNRole **Destination** *DIAL	**EventReleased** ConnID **1** ThisDN **B** ThirdPartyDN **C** OtherDN **A** CallState **Transferred** Cause **1stepTransfer**	**EventRinging** ConnID **1** ThisDN **C** OtherDN **A** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred**
		Answer (TAnswerCall)
		EventEstablished

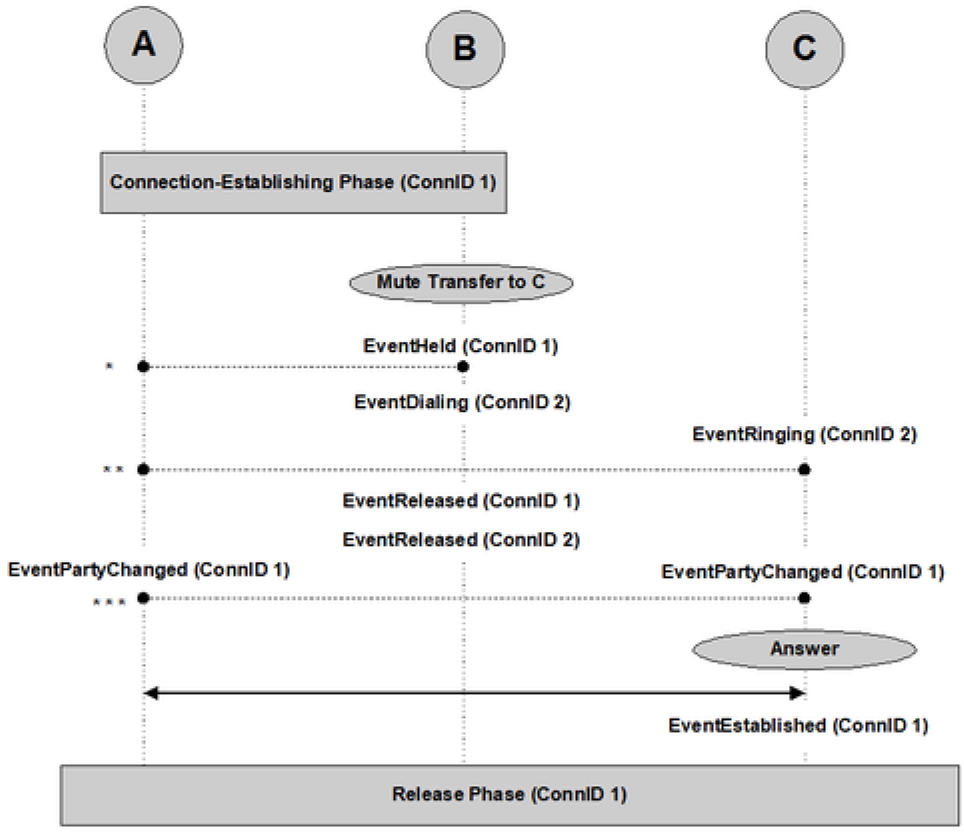
PARTY A	PARTY B	PARTY C
		ConnID **1** ThisDN **C** OtherDN **A**

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**

Mute Transfer

The following graphic and table describe a mute transfer.



Mute Transfer

PARTY A	PARTY B	PARTY C
	Call-Establishing Phase (ConnID 1)	
	Mute Transfer to C (TMuteTransfer*)	
	EventHeld ConnID **1** ThisDN **B** OtherDN **A**	
	EventDialing ConnID **2** ThisDN **B** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination**	**EventRinging** ConnID **2** ThisDN **C** ThisDNRole **Destination** OtherDN **B** OtherDNRole **Origination** CallState **OK**
EventPartyChanged ConnID **1** PreviousConnID **1**	**EventReleased** ConnID **1** ThisDN **B**	**EventPartyChanged** ConnID **1** PreviousConnID **2**

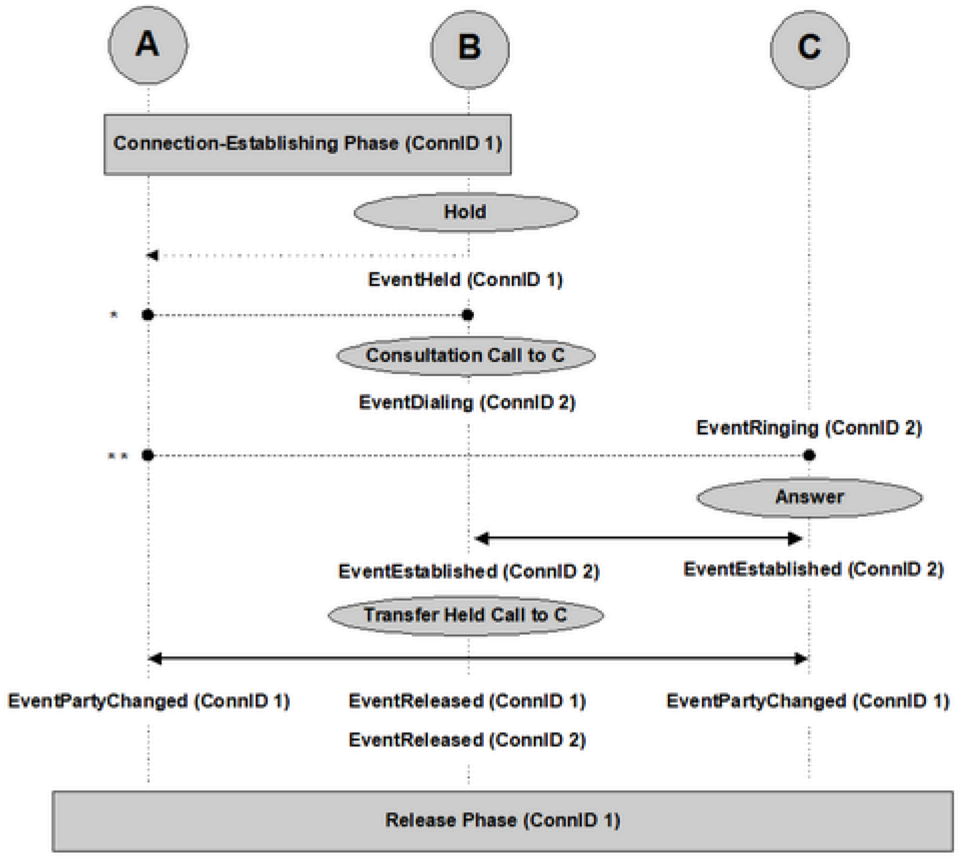
PARTY A	PARTY B	PARTY C
ThisDN **A** OtherDN **C** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred**	OtherDN **A** CallState **Transferred** **EventReleased** ConnID **2** ThisDN **B** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination** CallState **Transferred**	ThisDN **C** OtherDN **A** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred**
		Answer (TAnswerCall)
		EventEstablished ConnID **1** ThisDN **C** OtherDN **A**
Release Phase (ConnID 1)		

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK** **EventReleased** ConnID **2** ThisDN **B** OtherDN **C** CallState **OK**	**EventAbandoned** ConnID **2** ThisDN **C** OtherDN **B** CallState **OK**
***	**EventReleased** ConnID **1** ThisDN **B** OtherDN **C** CallState **OK**		**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **B** CallState **OK**

Two-Step Transfer: Complete After Consulted Party Answers

The following graphic and table describe a two-step transfer: complete after the consulted party answers.



Two-Step Transfer (Complete After Consulted Party Answers)

PARTY A	PARTY B	PARTY C
	Call-Establishing Phase (ConnID 1)	
	Hold (TInitiateTransfer*)	
	EventHeld ConnID **1** ThisDN **B** OtherDN **A**	
	Consultation Call to C **(TInitiateTransfer continues)**	

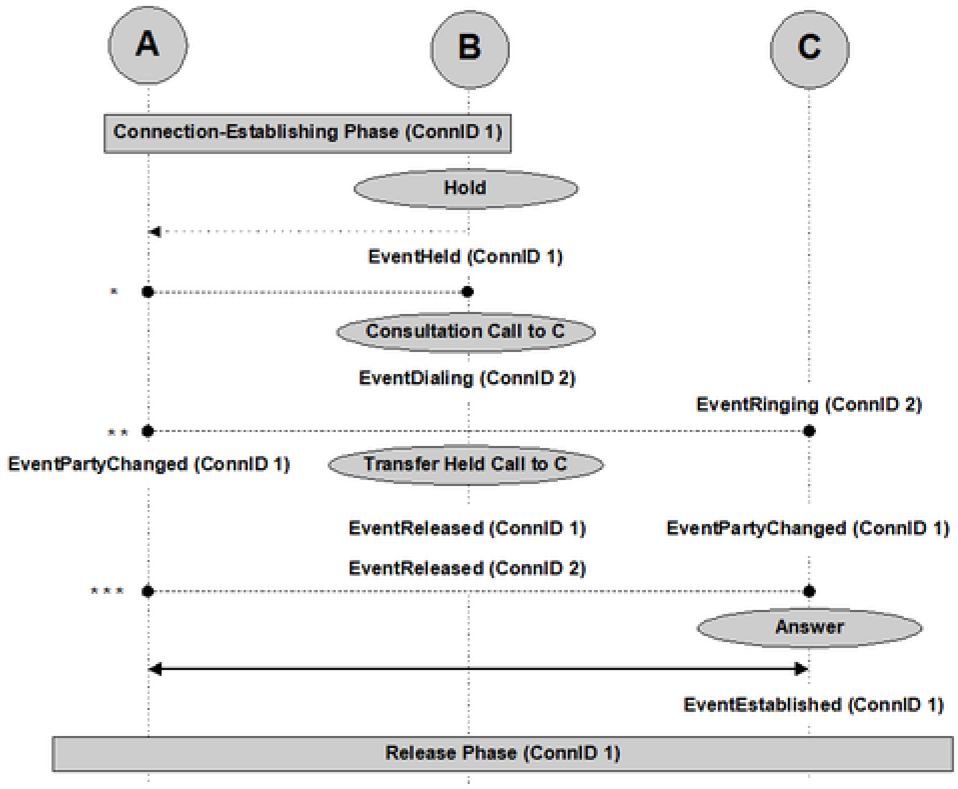
PARTY A	PARTY B	PARTY C
Call-Establishing Phase (ConnID 2)		
	Transfer Held Call to C (TCompleteTransfer)	
EventPartyChanged ConnID **1** PreviousConnID **1** ThisDN **A** OtherDN **C** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **Transferred** **EventReleased** ConnID **2** ThisDN **B** OtherDN **C** CallState **Transferred**	**EventPartyChanged** ConnID **1** PreviousConnID **2** ThisDN **C** OtherDN **A** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred**
Release Phase (ConnID 1)		

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK** **EventReleased** ConnID **2** ThisDN **B** OtherDN **C** CallState **OK**	**EventAbandoned** ConnID **2** ThisDN **C** OtherDN **B** CallState **OK**

Two-Step Transfer: Complete Before Consulted Party Answers (Blind)

The following graphic and table describe a two-step transfer: complete before the consulted party answers (blind).



Two-Step Transfer: Complete Before Consulted Party Answers (Blind)

PARTY A	PARTY B	PARTY C
	Call-Establishing Phase (ConnID 1)	
	Hold (TInitiateTransfer)	
	EventHeld ConnID **1** ThisDN **B** OtherDN **A**	
	Consultation Call to C **(TInitiateTransfer continues)**	
	EventDialing ConnID **2** ThisDN **B** ThisDNRole **Origination** OtherDN **C** *DIAL OtherDNRole **Destination** *DIAL	**EventRinging** ConnID **2** ThisDN **C** ThisDNRole **Destination** OtherDN **B** OtherDNRole **Origination** CallState **OK**
	Transfer Held Call to C	

PARTY A	PARTY B	PARTY C
	(TCompleteTransfer)	
EventPartyChanged ConnID **1** PreviousConnID **1** ThisDN **A** OtherDN **C** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **Transferred** **EventReleased** ConnID **2** ThisDN **B** OtherDN **C** CallState **Transferred**	**EventPartyChanged** ConnID **1** PreviousConnID **2** ThisDN **C** OtherDN **A** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred**
		Answer (TAnswerCall)
		EventEstablished ConnID **1** ThisDN **C** OtherDN **A**
Release Phase (ConnID 1)		

Important

If a call appears on the terminating party after transfer completion, the `ConnID` field of `EventRinging` is equal to the connection ID of the original call (`ConnID 1`), and `EventPartyChanged` is not generated.

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK** **EventReleased** ConnID **2** ThisDN **B**	**EventAbandoned** ConnID **2** ThisDN **C** OtherDN **B** CallState **OK**

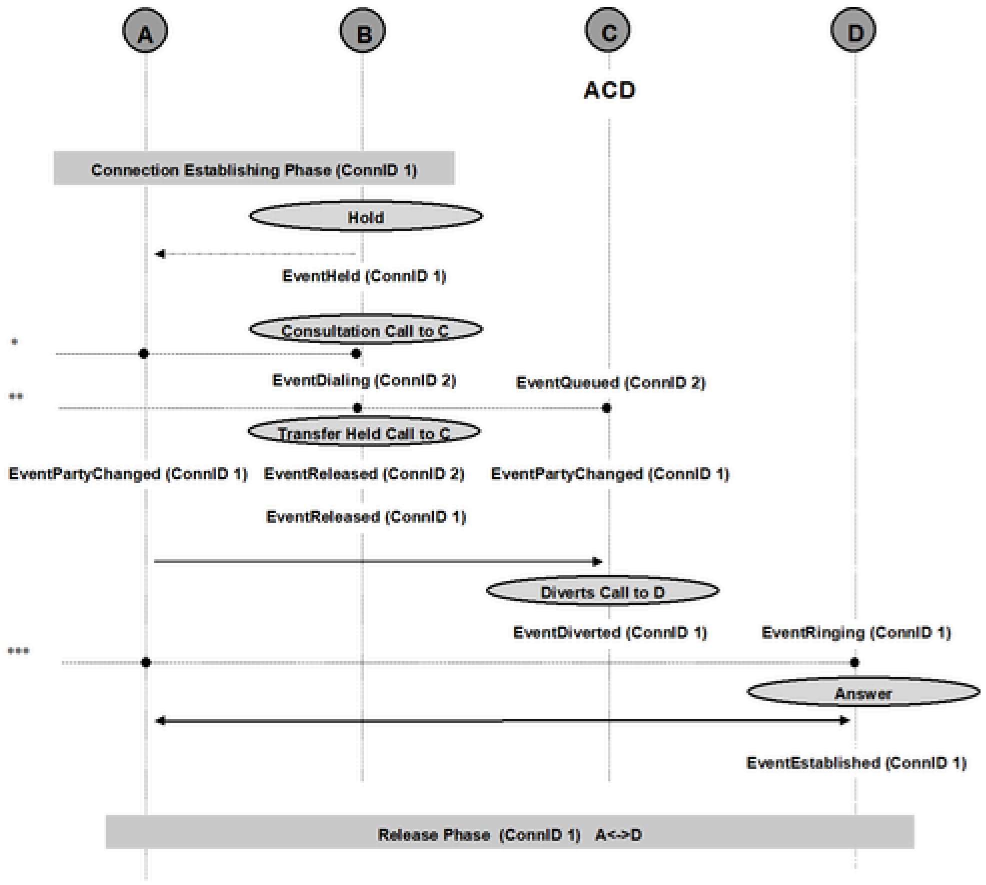
Interruption Point	PARTY A	PARTY B	PARTY C
		OtherDN **C** CallState **OK**	
***	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**

Two-Step Transfer to ACD

Important

Two-step transfer to ACD means that a call is waiting in a queue, and the transfer completed before any ACD agent is available to receive the call.

The following graphic and table describe a two-step transfer to ACD.



Two-Step Transfer to ACD

PARTY A	PARTY B	PARTY C (ACD)	PARTY D
Call-Establishing Phase (ConnID 1)			
	Hold (TInitiateTransfer)		
	EventHeld ConnID **1** ThisDN **B** OtherDN **A**		
	Consultation Call to C **(TInitiateTransfer continues)**		
	EventDialing ConnID **2** ThisDN **B** OtherDN **C** *DIAL	**EventQueued** ConnID **2** ThisDN **C** ThisQueue **C** OtherDN **B**	

PARTY A	PARTY B	PARTY C (ACD)	PARTY D
	Transfer Held Call to C (TCompleteTransfer)		
EventPartyChanged ConnID **1** PreviousConnID **1** ThisDN **A** OtherDN **C** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred**	**EventReleased** ConnID **2** ThisDN **B** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination** CallState **Transferred** **EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **Transferred**	**EventPartyChanged** ConnID **1** PreviousConnID **2** ThisDN **C** ThisQueue **C** OtherDN **A** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred**	
		Diverts Call to D	
		EventDiverted ConnID **1** ThisDN **C** OtherDN **A** ThirdPartyDN **C** *OPT ThirdPartyDNRole **Destination** *OPT	**EventRinging** ConnID **1** ThisDN **D** ThisQueue **C** OtherDN **A** CallState **OK**
			Answer (TAnswerCall)
			EventEstablished ConnID **1** ThisDN **D** ThisQueue **C** OtherDN **A** CallState **OK**
Release Phase (ConnID 1)			

Important

If a call transfer is completed before it is put in an ACD queue, an `EventPartyChanged` is not generated.

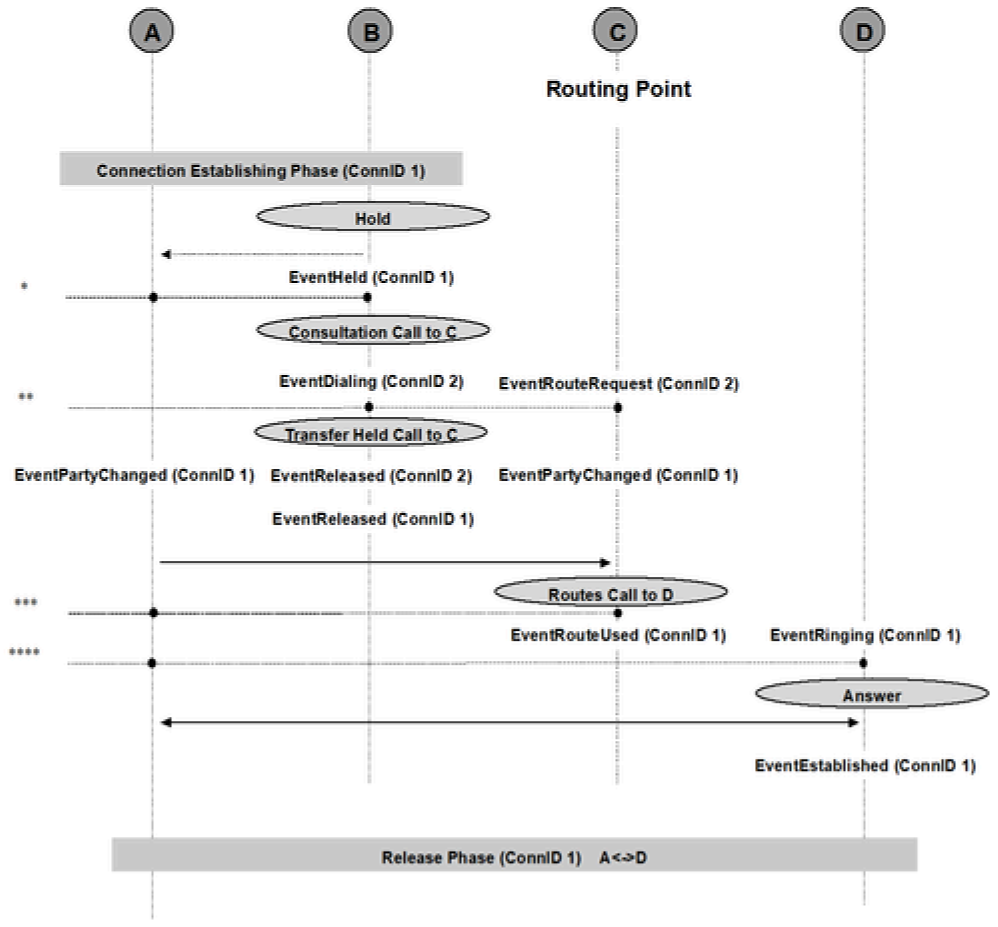
Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C	PARTY D
*	**EventReleased**	**EventReleased**		

Interruption Point	PARTY A	PARTY B	PARTY C	PARTY D
	ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**		
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK** **EventReleased** ConnID **2** ThisDN **B** OtherDN **C** CallState **OK**	**EventAbandoned** ConnID **2** ThisDN **C** OtherDN **B** CallState **OK**	
***	**EventReleased** ConnID **1** ThisDN **A** OtherDN **D** CallState **OK**			**EventAbandoned** ConnID **1** ThisDN **D** OtherDN **A** CallState **OK**

Two-Step Transfer to a Routing Point

The following graphic and table describe a two-step transfer to a routing point.



Two-Step Transfer to a Routing Point

PARTY A	PARTY B	PARTY C (ACD)	PARTY D
		Call-Establishing Phase (ConnID 1)	
	Hold (TInitiateTransfer)		
	EventHeld ConnID **1** ThisDN **B** OtherDN **A**		
	Consultation Call to C **(TInitiateTransfer continues)**		
	EventDialing	**EventRouteRequest**	

PARTY A	PARTY B	PARTY C (ACD)	PARTY D
	ConnID **2** ThisDN **B** ThisDNRole **Origination** OtherDN **C** *DIAL OtherDNRole **Destination** CallType **Consult**	ConnID **2** ThisDN **C** ThisDNRole **Destination** OtherDN **B** OtherDNRole **Origination**	
	Transfer Held Call to C **(TCompleteTransfer)**		
EventPartyChanged ConnID **1** PreviousConnID **1** ThisDN **A** ThisDNRole **Origination** [a] OtherDN **C** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred**	**EventReleased** ConnID **2** ThisDN **B** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination** CallState **Transferred** **EventReleased** ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** CallState **Transferred**	**EventPartyChanged** ConnID **1** PreviousConnID **2** ThisDN **C** OtherDN **A** ThirdPartyDN **B** ThirdPartyDNRole **TransferredBy** CallState **Transferred**	
		Diverts Call to D	
		EventRouteUsed ConnID **1** ThisDN **C** OtherDN **A** ThirdPartyDN **D** *OPT	**EventRinging** ConnID **1** ThisDN **D** OtherDN **A** CallState **OK**
			Answer (TAnswerCall)
			EventEstablished ConnID **1** ThisDN **D** OtherDN **A**
Call-Establishing Phase (ConnID 1)			

a. `ThisDNRole` must be Destination if party B is the call originator.

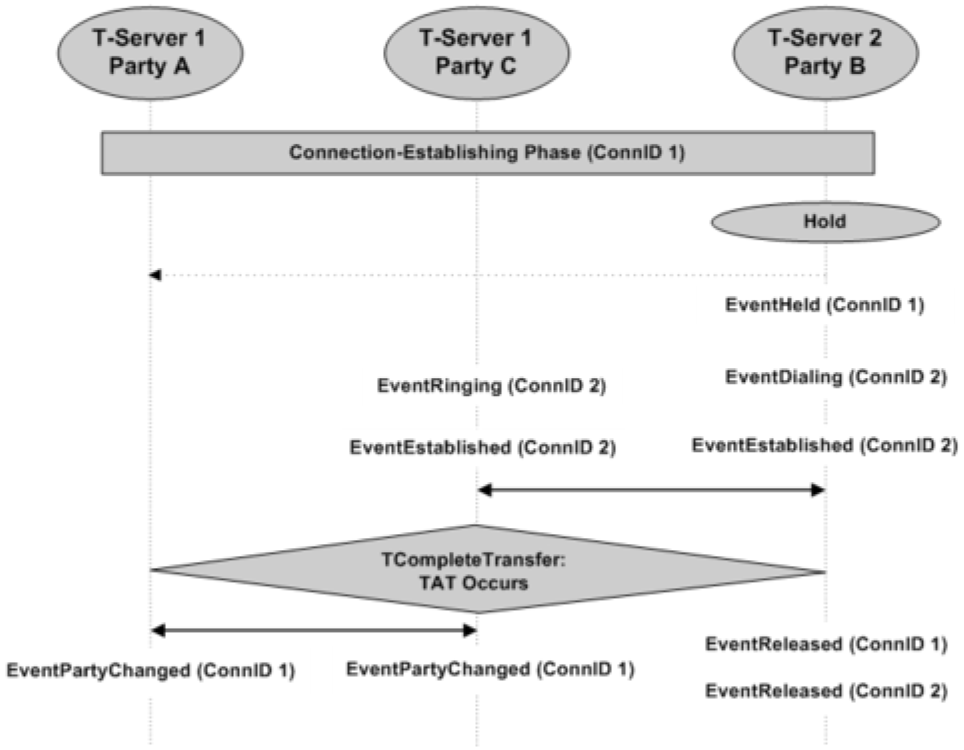
Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C	PARTY D
*	**EventReleased** ConnID **1**	**EventReleased** ConnID **1**		

Interruption Point	PARTY A	PARTY B	PARTY C	PARTY D
	ThisDN **A** OtherDN **B** CallState **OK**	ThisDN **B** OtherDN **A** CallState **OK**		
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK** **EventReleased** ConnID **2** ThisDN **B** OtherDN **C** CallState **OK**	**EventAbandoned** ConnID **2** ThisDN **C** OtherDN **B** CallState **OK**	
***	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**	
****	**EventReleased** ConnID **1** ThisDN **A** OtherDN **D** CallState **OK**			**EventAbandoned** ConnID **1** ThisDN **D** OtherDN **A** CallState **OK**

Trunk Optimization: Trunk Anti-Tromboning

Trunk optimization: trunk anti-tromboning (TAT) scenarios apply to functionality available from certain Nortel switches and are very similar to the case of Two-Step Transfer: Complete After Consulted Party Answers. The following graphic identifies the call model used by T-Servers to indicate a TAT event.



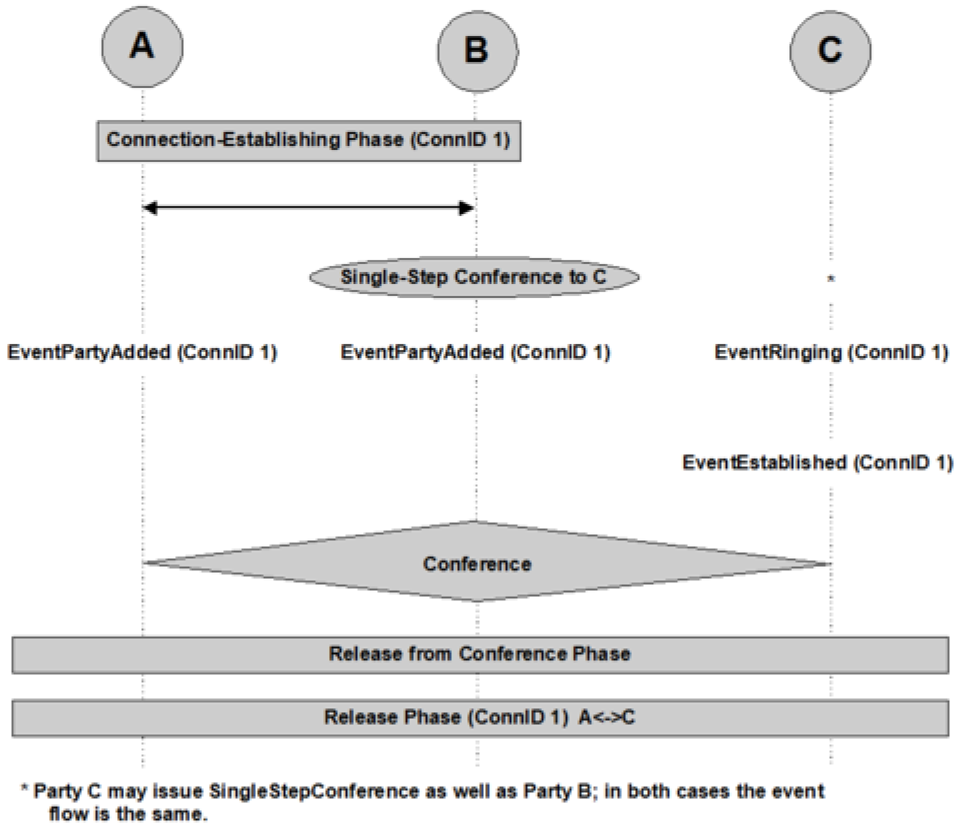
Trunk Optimization: Trunk Anti-Tromboning

T-Server 1 **PARTY A**	**T-Server 1** **PARTY C**	**T-Server 2** **PARTY B**
	Call-Establishing Phase (ConnID 1)	
		Hold (TInitiateTransfer to C)
		EventHeld ConnID **1** ThisDN **B** OtherDN **A**
	EventRinging ConnID **2** ThisDN **C** OtherDN **B**	**EventDialing** ConnID **2** ThisDN **B** OtherDN **C**
	EventEstablished ConnID **2** ThisDN **C** OtherDN **B**	**EventEstablished** ConnID **2** ThisDN **B** OtherDN **C**
		TCompleteTransfer

T-Server 1 **PARTY A**	**T-Server 1** **PARTY C**	**T-Server 2** **PARTY B**
	Trunk Optimization Occurs	
EventPartyChanged ConnID **1** ThisDN **A** OtherDN **C** ThirdPartyDN **B** CallState **RemoteRelease**	**EventPartyChanged** ConnID **1** PreviousConnID **2** ThisDN **C** OtherDN **A** ThirdPartyDN **B** CallState **RemoteRelease**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **Transferred** **EventReleased** ConnID **2** ThisDN **B** OtherDN **C** CallState **Transferred**

Single-Step Conference

The following graphic and table describe a single-step conference.


A
B
C
Connection-Establishing Phase (ConnID 1)
Single-Step Conference to C
*
EventPartyAdded (ConnID 1)
EventPartyAdded (ConnID 1)
EventRinging (ConnID 1)
EventEstablished (ConnID 1)
Conference
Release from Conference Phase
Release Phase (ConnID 1) A<->C
* Party C may issue SingleStepConference as well as Party B; in both cases the event flow is the same.

Single-Step Conference

PARTY A	PARTY B	PARTY C
Call-Establishing Phase (ConnID 1)		
	TSingleStepConference	
EventPartyAdded ConnID **1** ThisDN **A** OtherDN **C** ThirdPartyDN **B** [a]	**EventPartyAdded** ConnID **1** ThisDN **B** OtherDN **C** ThirdPartyDN **B** [a]	**EventRinging** ConnID **1** ThisDN **C** ThisDNRole **ConferenceMember** CallState **OK**
		EventEstablished ConnID **1** ThisDN **C** ThisDNRole **ConferenceMember** CallState **Conferenced**
Release from Conference Phase		
Release Phase (ConnID 1)		

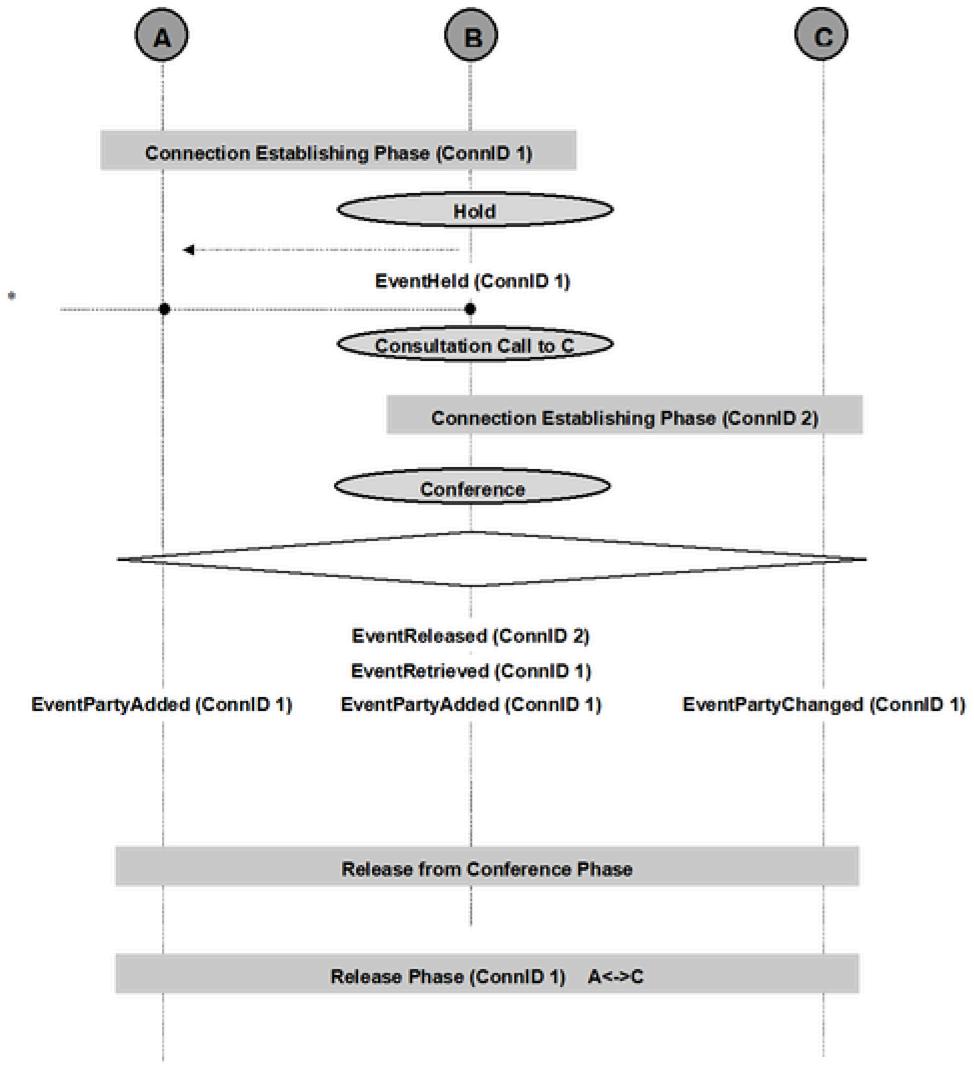
a. `ThirdPartyDN` has a value of C if Party C initiates the request for a conference.

Conference

The following graphic and table describe a conference.

Important

This call model applies to two types of conferences: Two-Step Conference and Conference with Calls Merge.


A
B
C
Connection Establishing Phase (ConnID 1)
Hold
EventHeld (ConnID 1)
Consultation Call to C
Connection Establishing Phase (ConnID 2)
Conference
EventReleased (ConnID 2)
EventRetrieved (ConnID 1)
EventPartyAdded (ConnID 1)
EventPartyAdded (ConnID 1)
EventPartyChanged (ConnID 1)
Release from Conference Phase
Release Phase (ConnID 1) A<->C

Conference

PARTY A	PARTY B	PARTY C
	Call-Establishing Phase (ConnID 1)	
	Hold (See Application Activities for Different Types of Conference.)	
	EventHeld ConnID **1** ThisDN **B** ThisDNRole **Previous Role of DN** OtherDN **A**	

PARTY A	PARTY B	PARTY C
	OtherDNRole **Previous Role of DN**	
	Consultation Call to C **(See Application Activities for Different Types of Conference.)**	
Call-Establishing Phase (ConnID 2)		
	Conference (See Application Activities for Different Types of Conference.)	
EventPartyAdded ConnID **1** ThisDN **A** OtherDN **C** ThirdPartyDN **B** ThirdPartyDNRole **AddedBy** CallState **Conferenced**	**EventReleased** ConnID **2** ThisDN **B** OtherDN **C** CallState **Conferenced** **EventRetrieved** [a] ConnID **1** ThisDN **B** OtherDN **A** CallState **Conferenced** **EventPartyAdded** ConnID **1** ThisDN **B** OtherDN [b] **C** OtherDNRole **NewParty** ThirdPartyDN **B** ThirdPartyDNRole **AddedBy** CallState **Conferenced**	**EventPartyChanged** ConnID **1** PreviousConnID **2** ThisDN **C** ThirdPartyDN **B** ThirdPartyDNRole **ConferencedBy** CallState **Conferenced**
Release from Conference Phase		
Release Phase (ConnID 1)		

a. With `EventRetrieved`, the values for attributes `ThisDNRole` and `ThisQueue` are the same as those for the attributes of the same names, if any, in the events preceding `EventRetrieved` (`EventEstablished` and `EvenRinging`). For non-ACD calls, however, `ThisQueue` is not reported. b. If only one party is added (as in the case of a simple conference call), the corresponding telephony object is specified in OtherDN. If more than one party is added, then the corresponding telephony objects are specified in `Extensions`.

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B
*	**EventReleased**	**EventReleased**

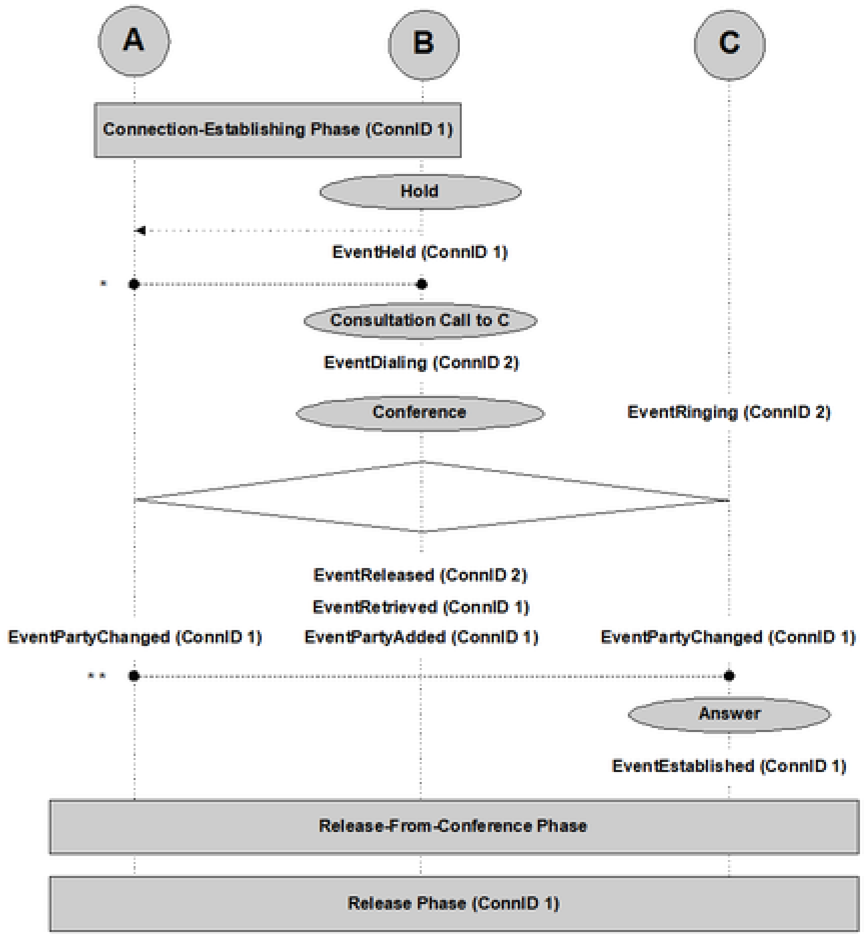
Interruption Point	PARTY A	PARTY B
	ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**

Application Activities for Different Types of Conference

Call Phase	Two-Step Conference	Conference with Calls Merge
HOLD	TInitiateConference	THoldCall
CONSULTATION CALL		TMakeCall
CONFERENCE	TCompleteConference	TMergeCalls

Blind Conference (Complete Before Consulted Party Answers)

The following graphic and table describe a blind conference (complete before the consulted party answers).


A
B
C
Connection-Establishing Phase (ConnID 1)
Hold
EventHeld (ConnID 1)
*
Consultation Call to C
EventDialing (ConnID 2)
Conference
EventRinging (ConnID 2)
EventReleased (ConnID 2)
EventRetrieved (ConnID 1)
EventPartyChanged (ConnID 1)
EventPartyAdded (ConnID 1)
EventPartyChanged (ConnID 1)
**
Answer
EventEstablished (ConnID 1)
Release-From-Conference Phase
Release Phase (ConnID 1)

Blind Conference (Complete Before Consulted Party Answers)

PARTY A	PARTY B	PARTY C
	Call-Establishing Phase (ConnID 1)	
	Hold (See Application Activities for Different Types of Conference.)	
	EventHeld ConnID **1** ThisDN **B** OtherDN **A**	
	Consultation Call to C **(See Application Activities for Different Types of Conference.)**	

PARTY A	PARTY B	PARTY C
	EventDialing ConnID **2** ThisDN **B** ThisDNRole **Origination** OtherDN **C** *DIAL OtherDNRole **Destination** *DIAL CallType **Consult**	**EventRinging** ConnID **2** ThisDN **C** ThisDNRole **Destination** OtherDN **B** OtherDNRole **Origination** CallType **Consult**
	Conference (See Application Activities for Different Types of Conference.)	
EventPartyAdded ConnID **1** ThisDN **A** OtherDN **C** ThirdPartyDN **B** ThirdPartyDNRole **AddedBy** CallState **Conferenced**	**EventReleased** ConnID **2** ThisDN **B** OtherDN **C** CallState **Conferenced** **EventRetrieved** [a] ConnID **1** ThisDN **B** OtherDN **A** CallState **Conferenced** **EventPartyAdded** ConnID **1** ThisDN **B** OtherDN **C** ThirdPartyDN **B** ThirdPartyDNRole **AddedBy** CallState **Conferenced**	**EventPartyChanged** ConnID **1** PreviousConnID **2** ThisDN **C** ThirdPartyDN **B** ThirdPartyDNRole **ConferencedBy** CallState **Conferenced**
		Answer (TAnswerCall)
		EventEstablished ConnID **1** ThisDN **C** CallState **Conferenced**
Release from Conference Phase		
Release Phase (ConnID 1)		

a. With `EventRetrieved`, the values for attributes `ThisDNRole` and `ThisQueue` are the same as those for the attributes of the same names, if any, in the events preceding `EventRetrieved` (`EventEstablished` and `EventRinging`). For non-ACD calls, however, `ThisQueue` is not reported.

Important

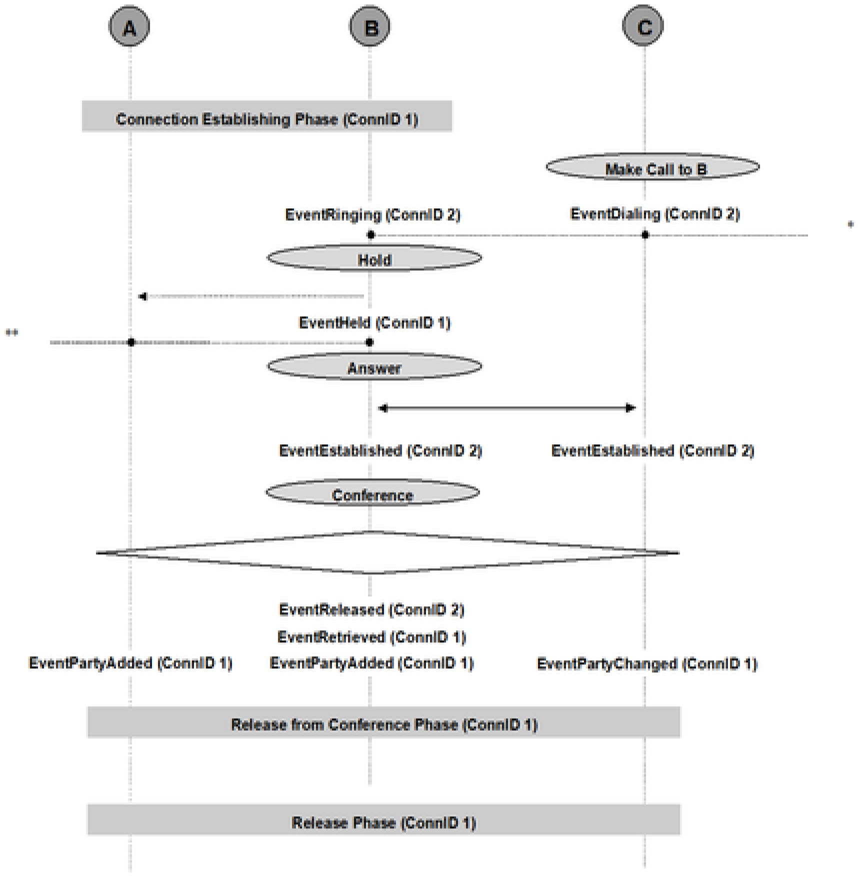
If a call appears on the terminating party after completion of conference, the `ConnID` field of `EventRinging` is equal to the connection ID of the original call (`ConnID 1`), and `EventPartyChanged` is not generated.

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** *DIAL CallState **OK**	**EventPartyDeleted** ConnID **1** ThisDN **B** OtherDN **A** OtherDNRole **DeletedParty** ThirdPartyDN **A** ThirdPartyDNRole **DeletedBy** CallState **OK**	

Conference with Two Incoming Calls Using TMergeCalls

The following graphic and table describe a conference with two incoming calls using `TMergeCalls`.


A
B
C
Connection Establishing Phase (ConnID 1)
Make Call to B
EventRinging (ConnID 2)
EventDialing (ConnID 2)
*
Hold
EventHeld (ConnID 1)
**
Answer
EventEstablished (ConnID 2)
EventEstablished (ConnID 2)
Conference
EventReleased (ConnID 2)
EventRetrieved (ConnID 1)
EventPartyAdded (ConnID 1)
EventPartyAdded (ConnID 1)
EventPartyChanged (ConnID 1)
Release from Conference Phase (ConnID 1)
Release Phase (ConnID 1)

Conference with Two Incoming Calls Using TMergeCalls

PARTY A	PARTY B	PARTY C
Call-Establishing Phase (ConnID 1)		
		Make Call to B (TMakeCall)
	EventRinging ConnID **2** ThisDN **B** ThisDNRole **Destination** OtherDN **C** OtherDNRole **Origination** CallState **OK**	**EventDialing** ConnID **2** ThisDN **C** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination**
	Place A on Hold (THoldCall)	
	EventHeld ConnID **1** ThisDN **B** OtherDN **A**	

PARTY A	PARTY B	PARTY C
	Answer (TAnswerCall)	
	EventEstablished ConnID **2** ThisDN **B** ThisDNRole **Destination** OtherDN **C** OtherDNRole **Origination**	**EventEstablished** ConnID **2** ThisDN **C** ThisDNRole **Origination** OtherDN **B** OtherDNRole **Destination**
	Conference (TMergeCalls)	
EventPartyAdded ConnID **1** ThisDN **A** OtherDN **C** OtherDNRole **NewParty** ThirdPartyDN **B** ThirdPartyDNRole **AddedBy** CallState **Conferenced**	**EventReleased** ConnID **2** ThisDN **B** ThisDNRole **Destination** OtherDN **C** OtherDNRole **Origination** CallState **Conferenced** **EventRetrieved** [a] ConnID **1** ThisDN **B** OtherDN **A** CallState **Conferenced** **EventPartyAdded** ConnID **1** ThisDN **B** OtherDN **C** OtherDNRole **NewParty** ThirdPartyDN **B** ThirdPartyDNRole **AddedBy** CallState **Conferenced**	**EventPartyChanged** ConnID **1** PreviousConnID **2** ThisDN **C** ThirdPartyDN **B** ThirdPartyDNRole **ConferencedBy** CallState **Conferenced**
	Release from Conference Phase	
	Release Phase (ConnID 1)	

a. With `EventRetrieved`, the values for attributes `ThisDNRole` and `ThisQueue` are the same as those for the attributes of the same names, if any, in the events preceding `EventRetrieved` (`EventEstablished` and `EventRinging`). For non-ACD calls, however, `ThisQueue` is not reported.

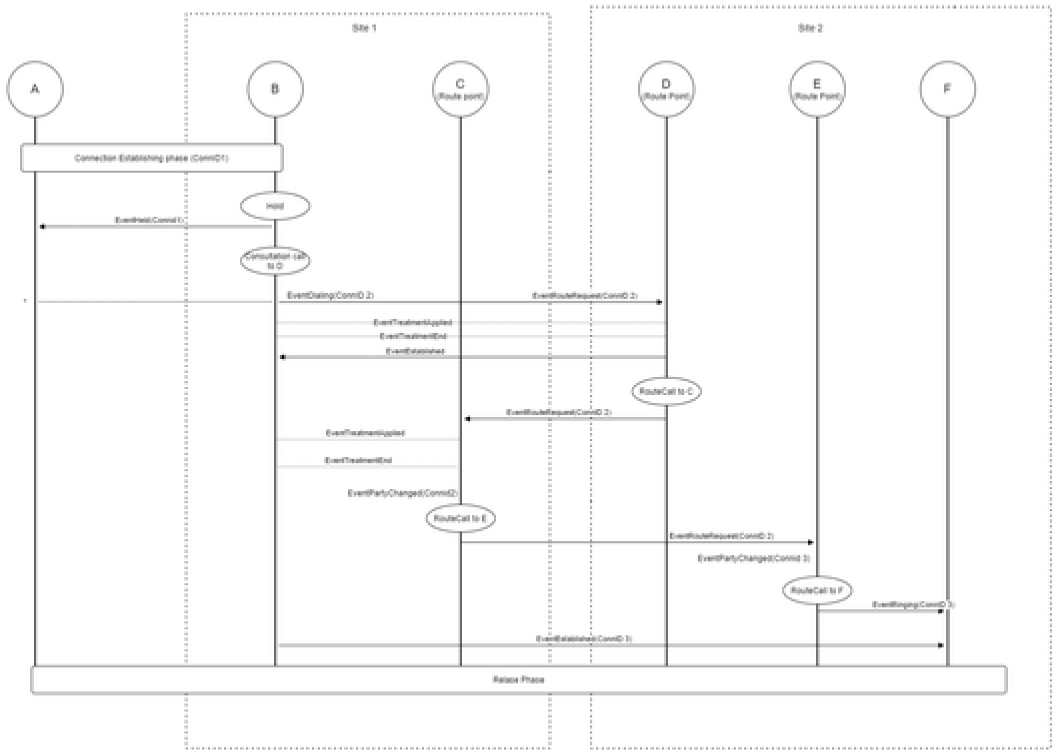
Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*		**EventAbandoned** ConnID **2** ThisDN **B** OtherDN **C** CallState **OK**	**EventReleased** ConnID **2** ThisDN **C** OtherDN **B** CallState **OK**
**	**EventReleased** ConnID **1**	**EventReleased** ConnID **1**	

Interruption Point	PARTY A	PARTY B	PARTY C
	ThisDN **A** OtherDN **B** CallState **OK**	OtherDN **A** CallState **OK**	

Special case: Multi-site ISCC Transfers and Conferences

The following graphic and table describe a case involving multi-site ISCC transfers and conferences, in which `EventPartyChanged` may contain `AttributeCallState` set to `0`.



Multi-site ISCC Transfers and Conferences

PARTY A	PARTY B	PARTY C	PARTY D	PARTY E	PARTY F
Call Establishing Phase					
	Hold (TInitiateTransfer)				
	EventHeld ConnID **1** ThisDN **B**				

PARTY A	PARTY B	PARTY C	PARTY D	PARTY E	PARTY F
	OtherDN **A**				
	Consultation call to D (TInitiateTransfer continues)				
	EventDialing ConnID **2** ThisDN **B** OtherDN **D**		**EventQueued** ConnID **2** ThisDN **D** ThisQueue **D** OtherDN **B**		
			TreatmentApplied at D		
	EventEstablished ConnID **2** ThisDN **B** OtherDN **D** CallState **OK**				
			Call Routed to C		
		EventQueued ConnID **2** ThisDN **C** ThisQueue **C** OtherDN **D**	**EventDiverted** ConnID **2** ThisDN **D** OtherDN **A** ThirdPartyDN **D** ThirdPartyDNRole **Destination**		
		TreatmentApplied at C			
		EventPartyChanged ConnID **2** ThisDN **B** OtherDN **D** CallState **Transferred**			
		Call Routed to E			
		EventDiverted ThisDN **C** OtherDN **A** ThirdPartyDN **E** ThirdPartyDNRole **Destination**		**EventQueued** ConnID **2** ThisDN **E** ThisQueue **E** OtherDN **C**	

PARTY A	PARTY B	PARTY C	PARTY D	PARTY E	PARTY F
		EventPartyChanged ThisDN **E** ThisDNRole **Destination** ConnID **3** PreviousConnectionID **2** CallState **0**		**EventPartyChanged** ThisDN **B** ThisDNRole **Origination** ConnID **3** PreviousConnectionID **2** CallState **0**	
				Call Routed to F	
				EventDiverted ThisDN **E** OtherDN **B** ConnID **3** ThirdPartyDN **F** ThirdPartyDNRole **Destination**	**EventRinging** ThisDN **F** OtherDN **B** ConnID **3** CallState **OK**
					EventEstablished ThisDN **F** OtherDN **B** ConnID **3**
Completion/Releasing phase					

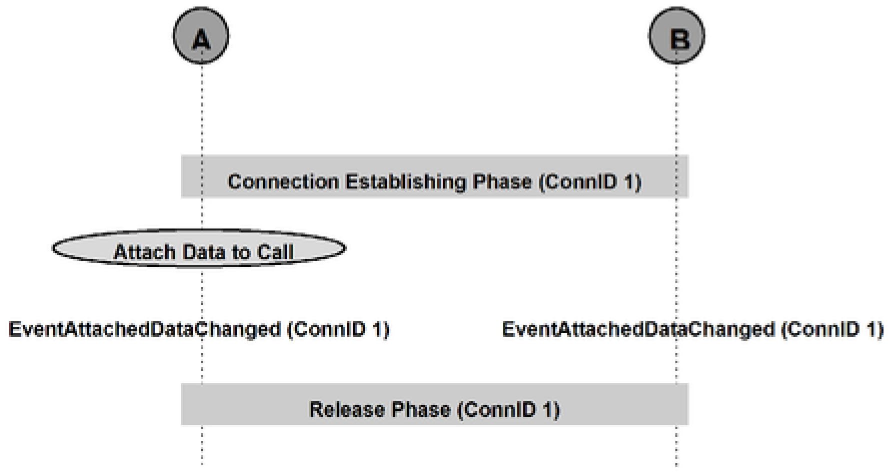
Abnormal Call Flow

Interruption Point	PARTY B	PARTY D	PARTY C	PARTY F
*	**EventReleased** ConnID **1** ThisDN **B** OtherDN **D** CallState **OK**	**EventReleased** ConnID **1** ThisDN **D** OtherDN **B** CallState **OK**		

Handling User Data

Attaching/Updating User Data to Internal Call

The following graphic and table describe attaching/updating user data to an internal call.

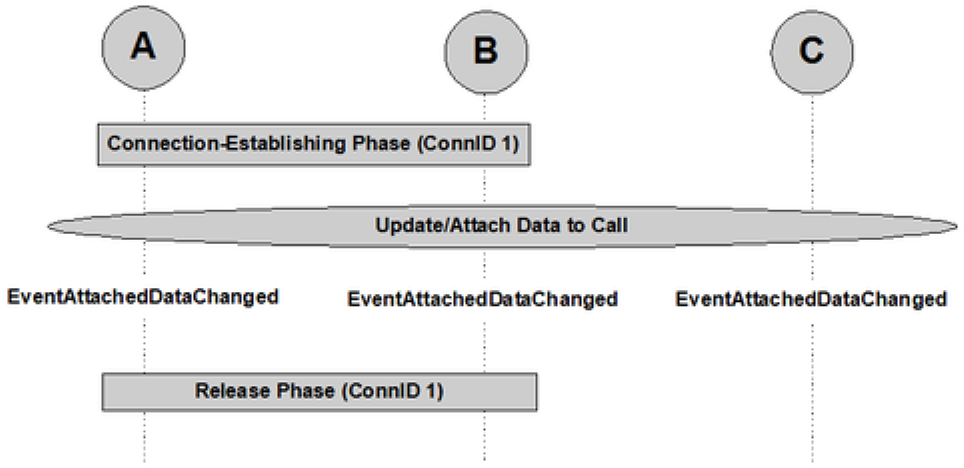

A
B
Connection Establishing Phase (ConnID 1)
Attach Data to Call
EventAttachedDataChanged (ConnID 1)
EventAttachedDataChanged (ConnID 1)
Release Phase (ConnID 1)

Attaching/Updating User Data to Internal Call

PARTY A	PARTY B
Call-Establishing Phase (ConnID 1)	
Attach User Data to a Call **(TUpdateUserData)**	
EventAttachedDataChanged ConnID **1** ThisDN **A** ThirdPartyDN **A**	**EventAttachedDataChanged** ConnID **1** ThisDN **B** ThirdPartyDN **A**
Release Phase (ConnID 1)	

Attaching/Updating User Data to Call by Third Party

The following graphic and table describe attaching/updating user data to a call by a third party.



Attaching/Updating User Data to Call by Third Party

PARTY A	PARTY B	PARTY C
	Call-Establishing Phase (ConnID 1)	
		Attach User Data to a Call **(TUpdateUserData)**
EventAttachedDataChanged ConnID **1** ThisDN **A** ThirdPartyDN **C**	**EventAttachedDataChanged** ConnID **1** ThisDN **B** ThirdPartyDN **C**	**EventAttachedDataChanged** ConnID **1** ThirdPartyDN **C**
	Release Phase (ConnID 1)	

Special Cases

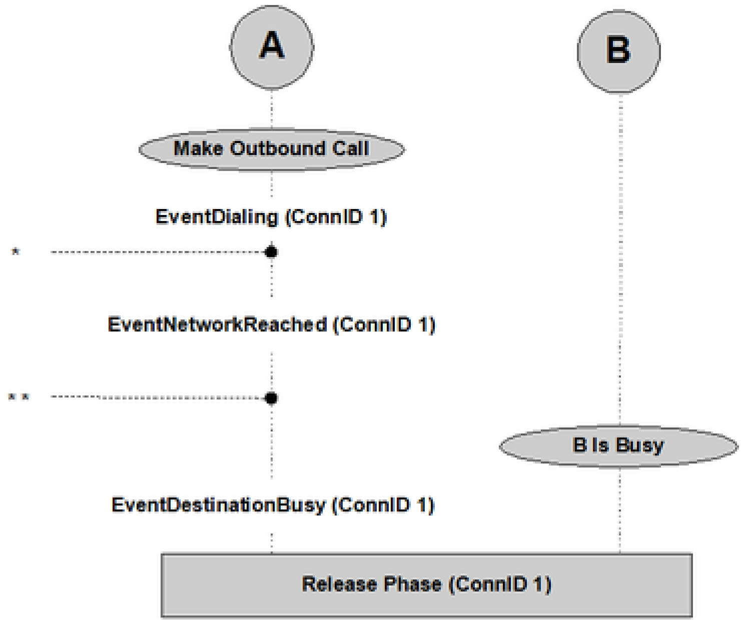
Note the following comments in the call models:

*OPT—Optional.

*DIAL—May be a dialed number or is not present if T-Server has no information about the other party.

Outbound Call to a Busy Destination

The following graphic and table describe an outbound call to a busy destination.



Outbound Call to a Busy Destination

PARTY A	PARTY B
Make Outbound Call to B (TMakeCall)	
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL **EventNetworkReached**	

PARTY A	PARTY B
ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	
	B is busy
EventDestinationBusy ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	
Release Phase (ConnID 1)	

Abnormal Call Flow

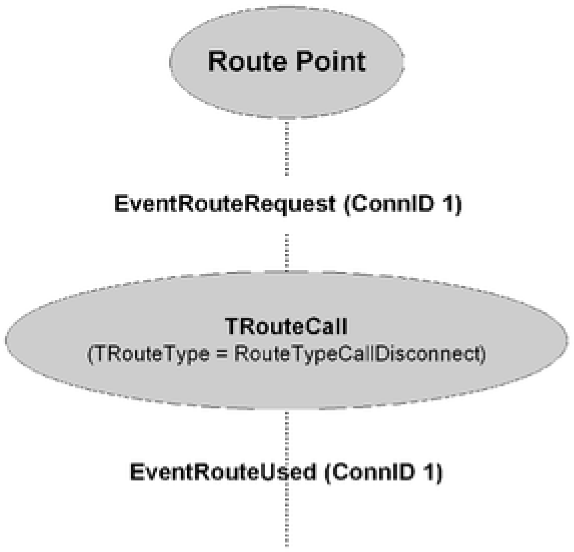
Interruption Point	PARTY A
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**

Rejected Call

Call rejection can apply both to incoming and outgoing calls. However, since most call centers forbid dropping the caller (without explaining why the call cannot be answered), for the inbound version of this, rejection is primarily for re-routing calls on a network level.

Generally, the rejected call scenario works either with `RouteTypeDefault` and an empty destination to reject the route request (using the default route destination as configured on the switch), or `RouteTypeCallDisconnect` to reject the call. (`RouteTypeReject` has been deprecated since it is switch-specific.) Two scenarios are applicable here. Rejected Call (Route Point) shows this with a route point involved, and Rejected Call (Route Queue) shows it with a route queue.

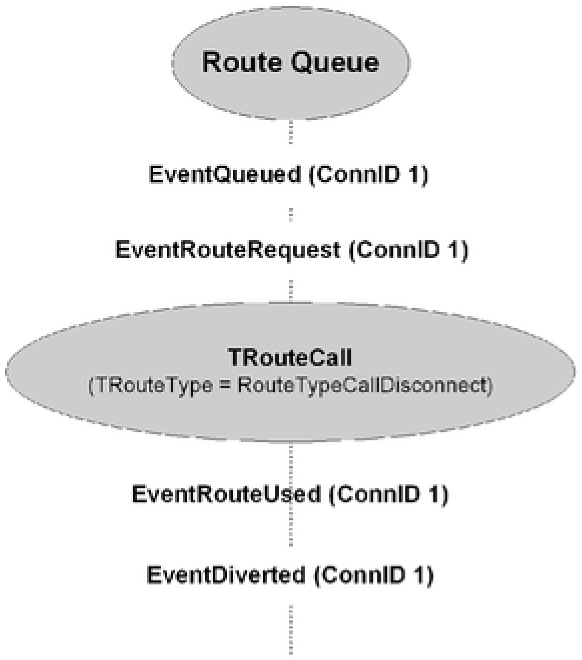
The following graphic and table describe a rejected call (route point).


Route Point
EventRouteRequest (ConnID 1)
TRouteCall
(TRouteType = RouteTypeCallDisconnect)
EventRouteUsed (ConnID 1)

Rejected Call (Route Point)

External Party	Route Point
Place Inbound Call to Route Point	
	EventRouteRequest ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**
	TRouteCall **(TRouteType=TRouteCallDisconnect)**
	EventRouteUsed ConnID **1** ThisDN **B** OtherDN **A** CallState **OK** **Note:** ThirdPartyDN is not present for this event.

The following graphic and table describe a rejected call (route queue).


Route Queue
EventQueued (ConnID 1)
EventRouteRequest (ConnID 1)
TRouteCall
(TRouteType = RouteTypeCallDisconnect)
EventRouteUsed (ConnID 1)
EventDiverted (ConnID 1)

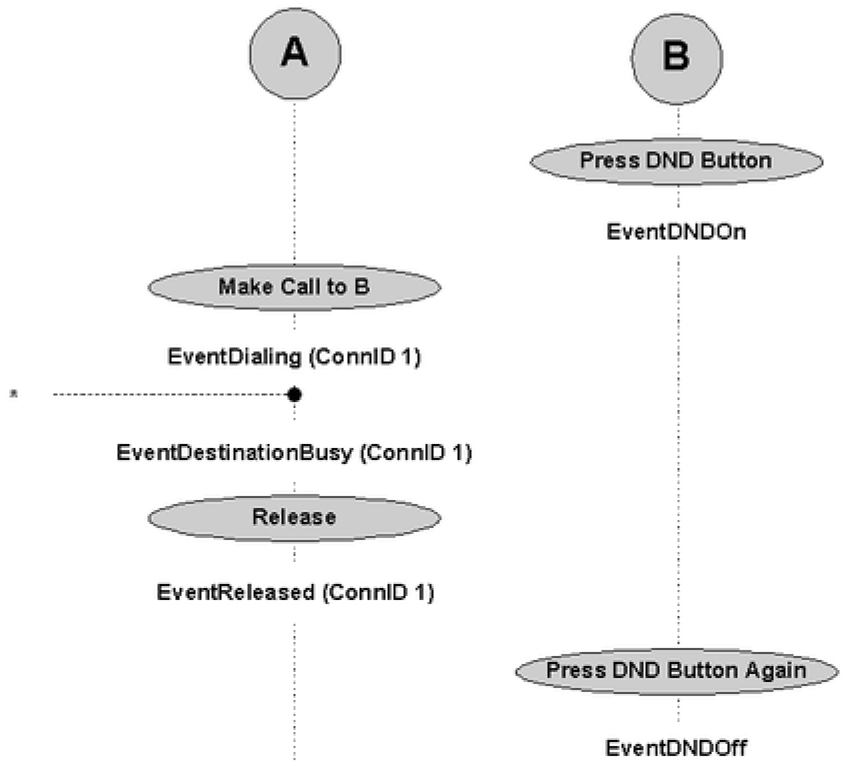
Rejected Call (Route Queue)

External Party	Route Queue
Place Inbound Call to Route Point	
	EventQueued ConnID **1** ThisDN **B** OtherDN **A** CallState **OK** **EventRouteRequest** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**
	TRouteCall **(TRouteType=TRouteCallDisconnect)**
	EventRouteUsed ConnID **1** ThisDN **B** OtherDN **A** CallState **OK** **Note:** ThirdPartyDN is not present for this event. **EventDiverted** ConnID **1** ThisDN **B** OtherDN **A**

External Party	Route Queue
	CallState **Dropped** **Note:** ThirdPartyDN is not present for this event.

Internal Call to Destination with DND Activated

The following graphic and table describe an internal call to destination with DND activated.



Internal Call to Destination with DND Activated

PARTY A	PARTY B
	Press DND button (TSetDNDOn)
	EventDNDOn ThisDN **B**
Make Call to B (TMakeCall)	
EventDialing	

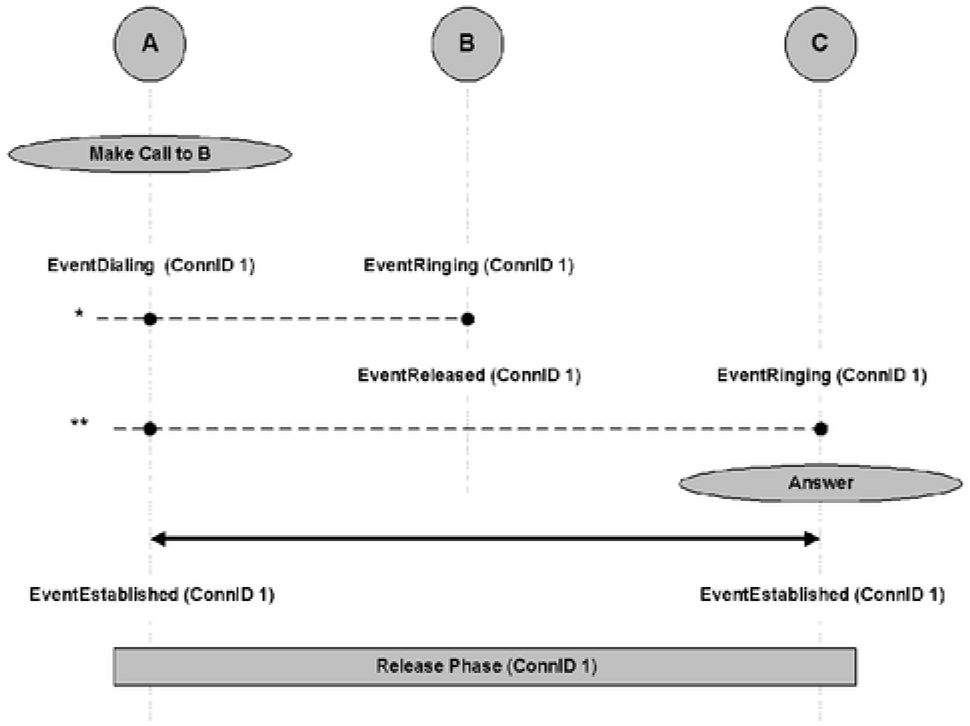
PARTY A	PARTY B
ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL **EventDestinationBusy** ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	
Release (TReleaseCall)	
EventReleased ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** OtherDNRole **Destination** CallState **OK**	
	Press DND button again (TSetDNDOff)
	EventDNDOff ThisDN **B**

Abnormal Call Flow

Interruption Point	PARTY A
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**

Call Forwarding (on No Answer)

The following graphic and table describe call forwarding (on no answer).



Call Forwarding (on No Answer)

PARTY A	PARTY B	PARTY C
Make Call to B (TMakeCall)		
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination**	**EventRinging** ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**	
	Call Forwarding **(On No Answer)**	
	EventReleased ConnID **1** ThisDN **B** ThirdPartyDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **Forwarded**	**EventRinging** ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **Forwarded**
		Answer (TAnswerCall)
EventEstablished		**EventEstablished**

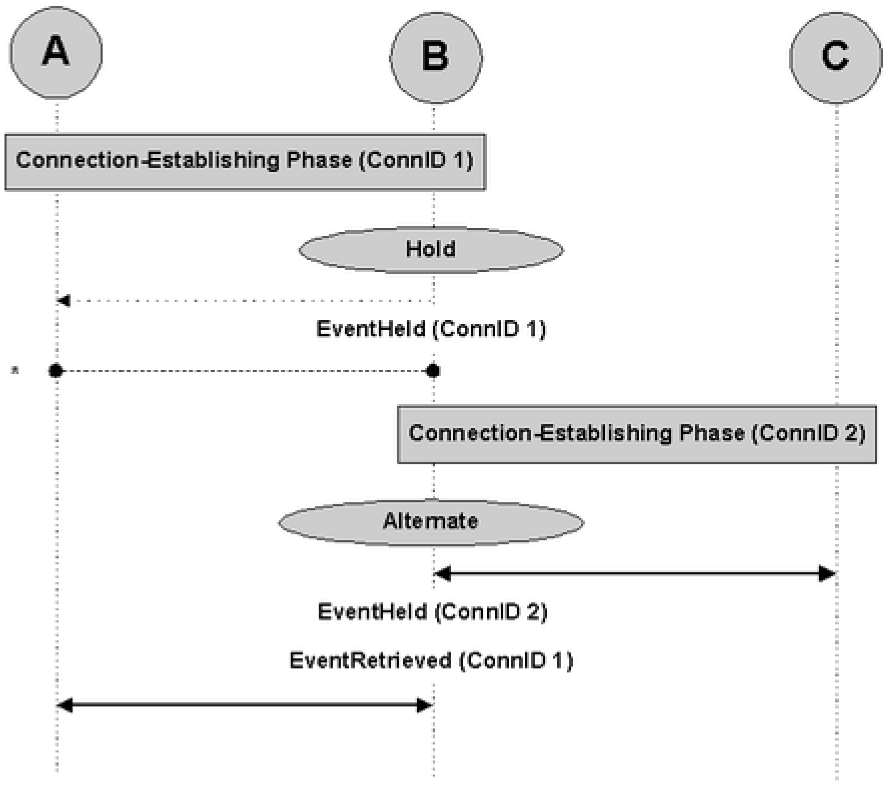
PARTY A	PARTY B	PARTY C
ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination**		ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**
Release Phase (ConnID 1)		

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**

Alternate-Call Service

The following graphic and table describe alternate-call service.


A
B
C
Connection-Establishing Phase (ConnID 1)
Hold
EventHeld (ConnID 1)
Connection-Establishing Phase (ConnID 2)
Alternate
EventHeld (ConnID 2)
EventRetrieved (ConnID 1)

Alternate-Call Service

PARTY A	PARTY B	PARTY C
	Call-Establishing Phase (ConnID 1)	
	Hold (THoldCall)	
	EventHeld ConnID **1** ThisDN **B** OtherDN **A**	
	Call-Establishing Phase (ConnID 2)	
	Alternate (TAlternateCall)	
	EventHeld ConnID **2** ThisDN **B** OtherDN **C**	
	EventRetrieved [a] ConnID **1**	

PARTY A	PARTY B	PARTY C
	ThisDN **B** OtherDN **A** CallState **OK**	
	Conversation (ConnID 1)	

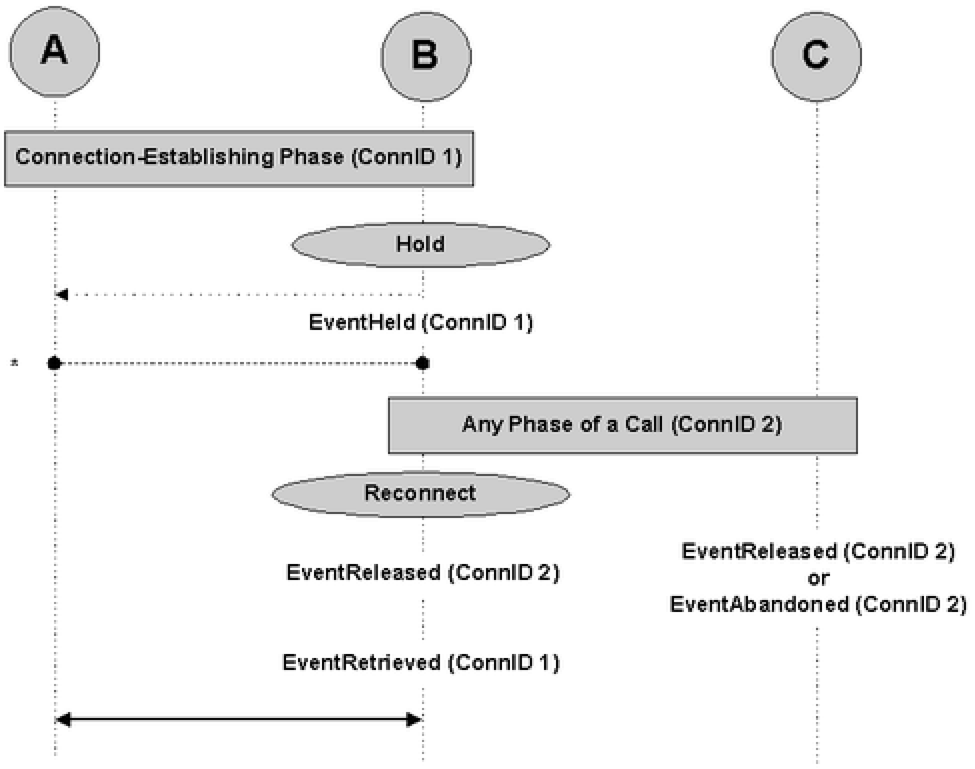
a. With `EventRetrieved`, the values for attributes `ThisDNRole` and `ThisQueue` are the same as those for the attributes of the same names, if any, in the events preceding `EventRetrieved` (`EventEstablished` and `EvenRinging`). For non-ACD calls, however, `ThisQueue` is not reported.

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	

Reconnect-Call Service

The following graphic and table describe reconnect-call service.


A
B
C
Connection-Establishing Phase (ConnID 1)
Hold
EventHeld (ConnID 1)
*
Any Phase of a Call (ConnID 2)
Reconnect
EventReleased (ConnID 2)
EventReleased (ConnID 2)
or
EventAbandoned (ConnID 2)
EventRetrieved (ConnID 1)

Reconnect-Call Service

PARTY A	PARTY B	PARTY C
	Call-Establishing Phase (ConnID 1)	
	Hold (THoldCall) or **Transfer (TInitiateTransfer) */ Conference (TInitiateConference)** [a]	
	EventHeld ConnID **1** ThisDN **B** OtherDN **A**	
	Any Phase of a Call (ConnID 2)	
	Reconnect (TReconnectCall)	
	EventReleased ConnID **2** ThisDN **B** OtherDN **C** CallState **OK** **EventRetrieved** [b] ConnID **1** ThisDN **B**	**EventReleased/ EventAbandoned** ConnID **2** ThisDN **C** OtherDN **B** CallState **OK**

PARTY A	PARTY B	PARTY C
	OtherDN **A** CallState **OK**	
Conversation (ConnID 1)		

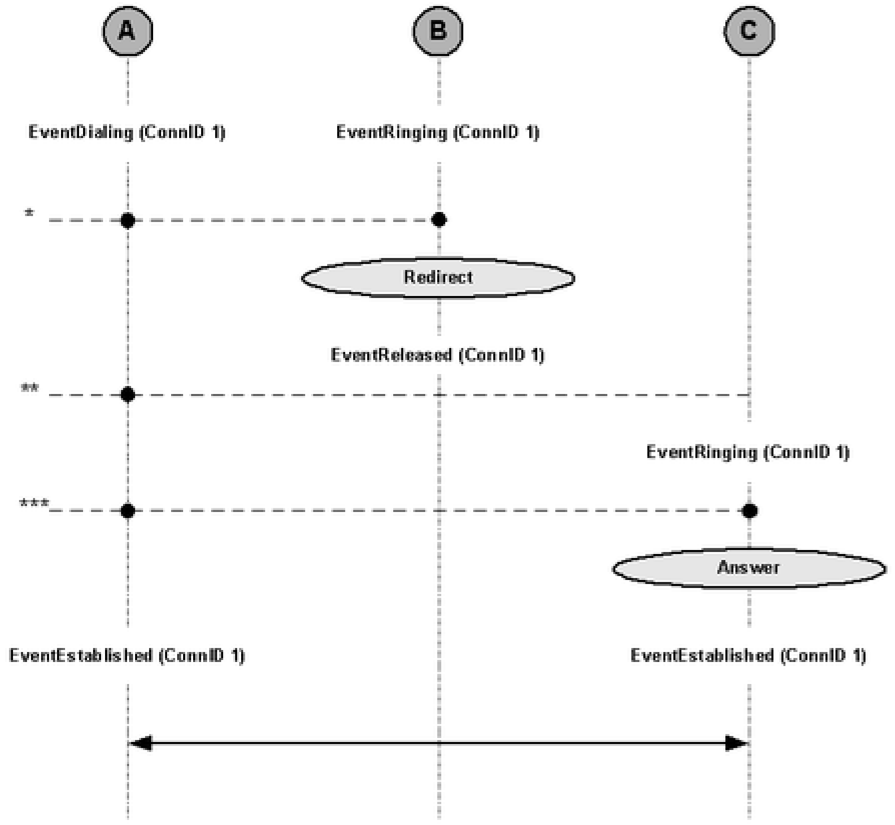
a. For the Hicom 300 E CS switch: service is available when `EventHeld` is generated as a result of one of these requests.
b. With `EventRetrieved`, the values for attributes `ThisDNRole` and `ThisQueue` are the same as those for the attributes of the same names, if any, in the events preceding `EventRetrieved` (`EventEstablished` and `EvenRinging`). For non-ACD calls, however, `ThisQueue` is not reported.

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	

Redirect-Call Service

The following graphic and table describe redirect-call service.



Redirect-Call Service

PARTY A	PARTY B	PARTY C
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	**EventRinging** ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**	
	Redirect (TRedirectCall)	
	EventReleased ConnID **1** ThisDN **B** ThirdPartyDN **C** OtherDN **A** CallState **Redirected**	**EventRinging** ConnID **1** ThisDN **C** ThirdPartyDN **B** CallState **Redirected**
		Answer (TAnswerCall)
EventEstablished		**EventEstablished**

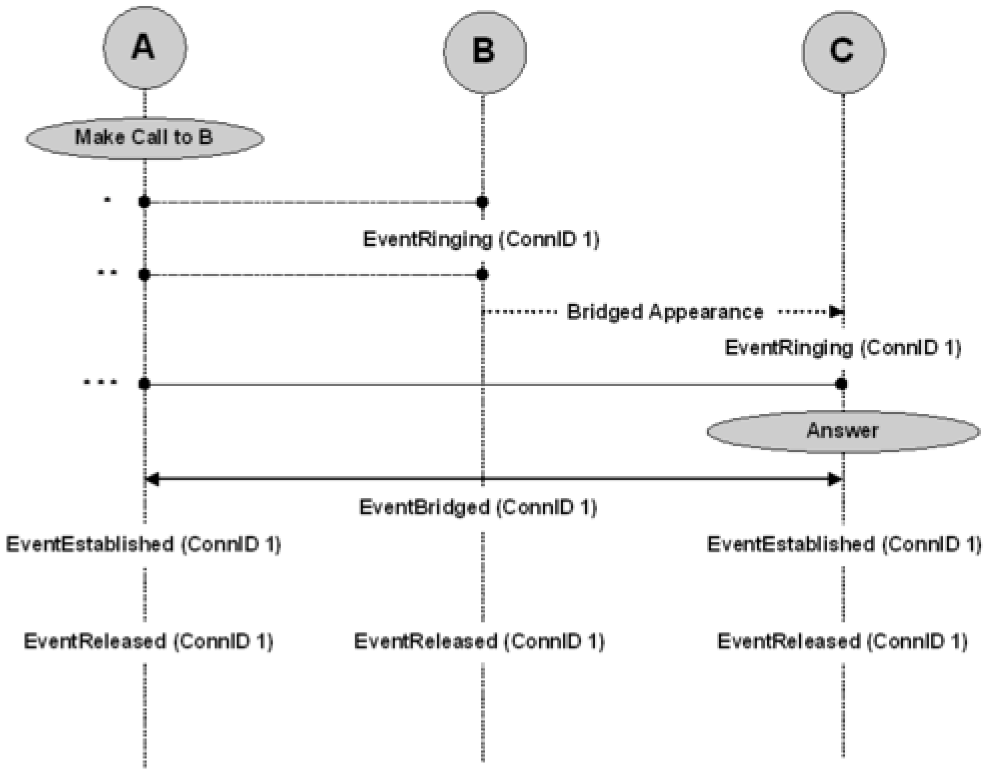
PARTY A	PARTY B	PARTY C
ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination**		ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **C** OtherDNRole **Origination**
Conversation (ConnID 1)		

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**		
***	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**

Internal/Inbound Call with Bridged Appearance

The following graphic and table describe an internal/inbound call with bridged appearance.



Internal/Inbound Call with Bridged Appearance

PARTY A	PARTY B	PARTY C
Make Call to B (TMakeCall) **or Inbound Call**		
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	**EventRinging** ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**	
	Coverage Path	
		EventRinging ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **Covered**
		Answer (TAnswerCall)
EventEstablished	**EventBridged**	**EventEstablished**

PARTY A	PARTY B	PARTY C
ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination**	ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**	ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**
Release [a]		**Release** [a]
EventReleased ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	**EventReleased** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**

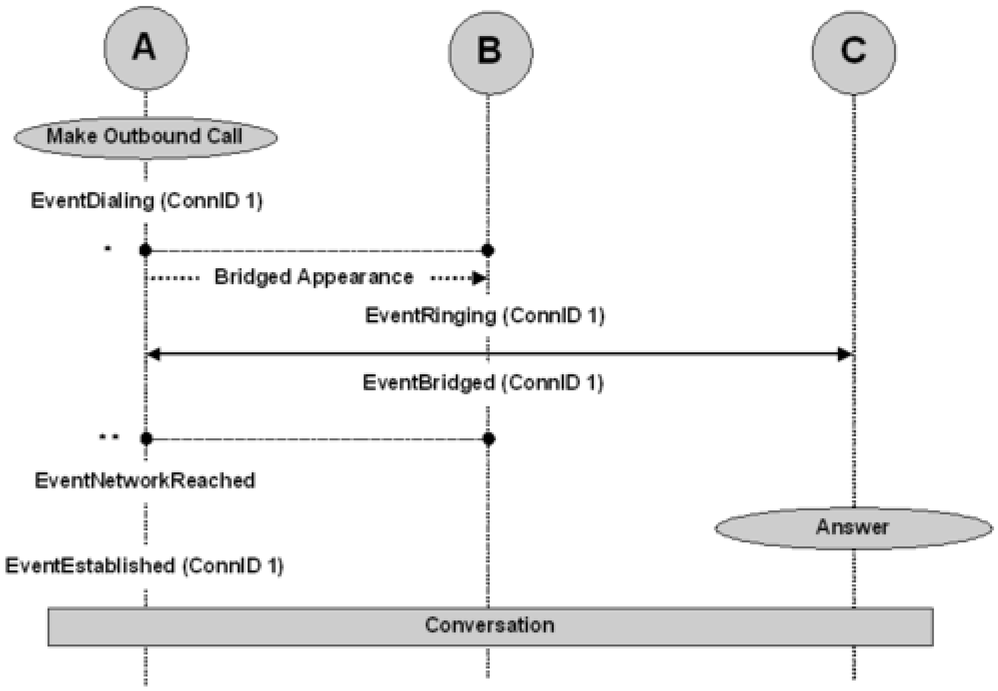
a. Either Party A or Party C can release the call.

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**		
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
***	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**

Outbound Call from Bridged Appearance

The following graphic and table describe an outbound call with bridged appearance.



Outbound Call from Bridged Appearance

PARTY A	PARTY B	PARTY C
Make Outside Call **(TMakeCall)**		
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** *DIAL OtherDNRole **Destination** *DIAL		
	EventRinging ConnID **1** ThisDN **B** ThisDNRole **Origination** OtherDN **C** *DIAL OtherDNRole **Destination** *DIAL CallState **Covered**	
	EventBridged ConnID **1** ThisDN **B** ThisDNRole **Origination** OtherDN **C** *OPT OtherDNRole **Destination**	

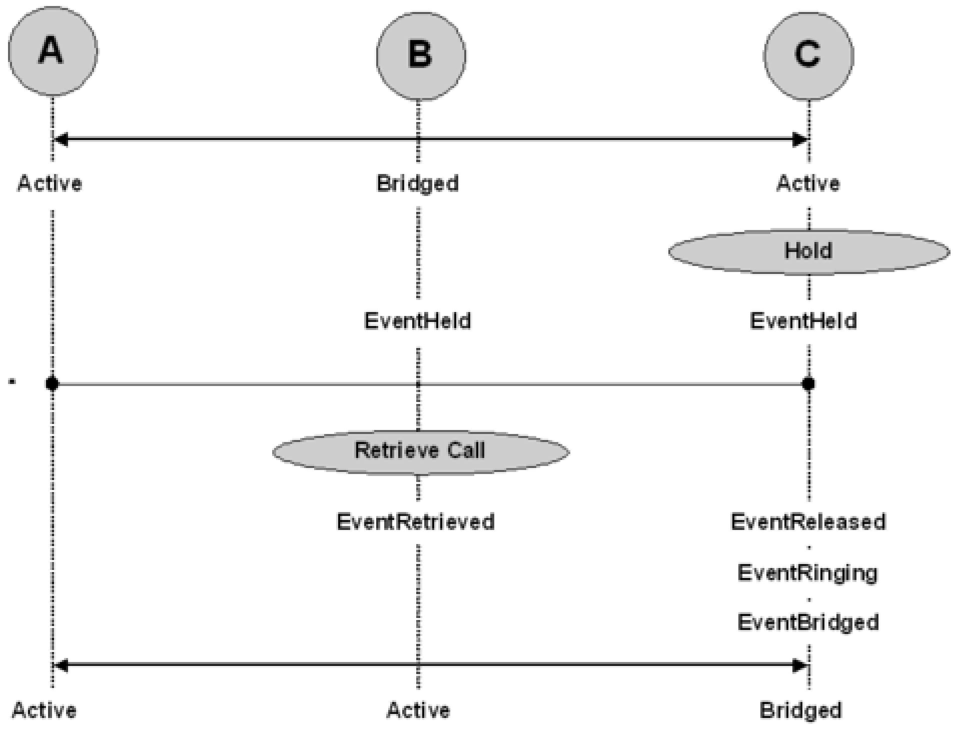
PARTY A	PARTY B	PARTY C
EventNetworkReached ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** *DIAL OtherDNRole **Destination** *DIAL		
		Answer
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** *OPT OtherDNRole **Destination** *OPT		

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **C** CallState **OK**	

Hold/Retrieve for Bridged Appearance

The following graphic and table describe hold/retrieve for bridged appearance.



Hold/Retrieve for Bridged Appearance

PARTY A	PARTY B	PARTY C
Call Established between Party A and Party C, with Party B Bridged		
		Hold (THoldCall)
	EventHeld ConnID **1** ThisDN **B** OtherDN **A**	**EventHeld** ConnID **1** ThisDN **C** OtherDN **A**
	Retrieve Call (TRetrieveCall)	
	EventRetrieved ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	**EventReleased** ConnID **1** ThisDN **C** OtherDN **A** CallState **Bridged** **EventRinging** ConnID **1** ThisDN **C** OtherDN **A** CallState **Covered** **EventBridged** ConnID **1** ThisDN **C**

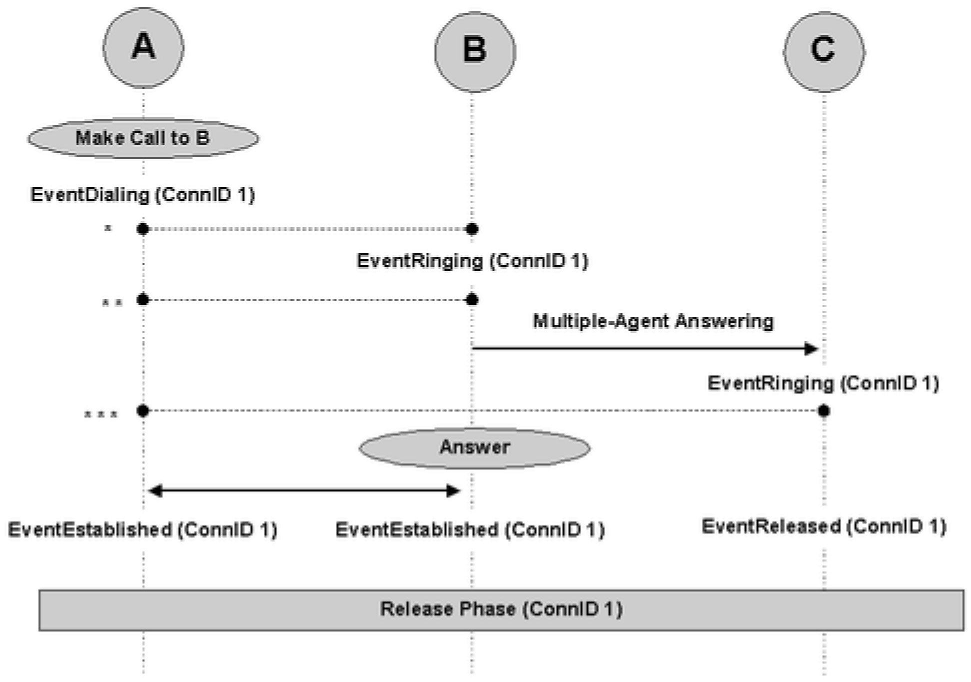
PARTY A	PARTY B	PARTY C
		OtherDN **A**

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** or **C** CallState **OK**	**EventReleased** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	**EventReleased** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**

Internal/Inbound Call Answerable by Several Agents (Party B Answers)

The following graphic and table describe an internal/inbound call answerable by several agents (Party B answers).



Internal/Inbound Call Answerable by Several Agents (Party B Answers)

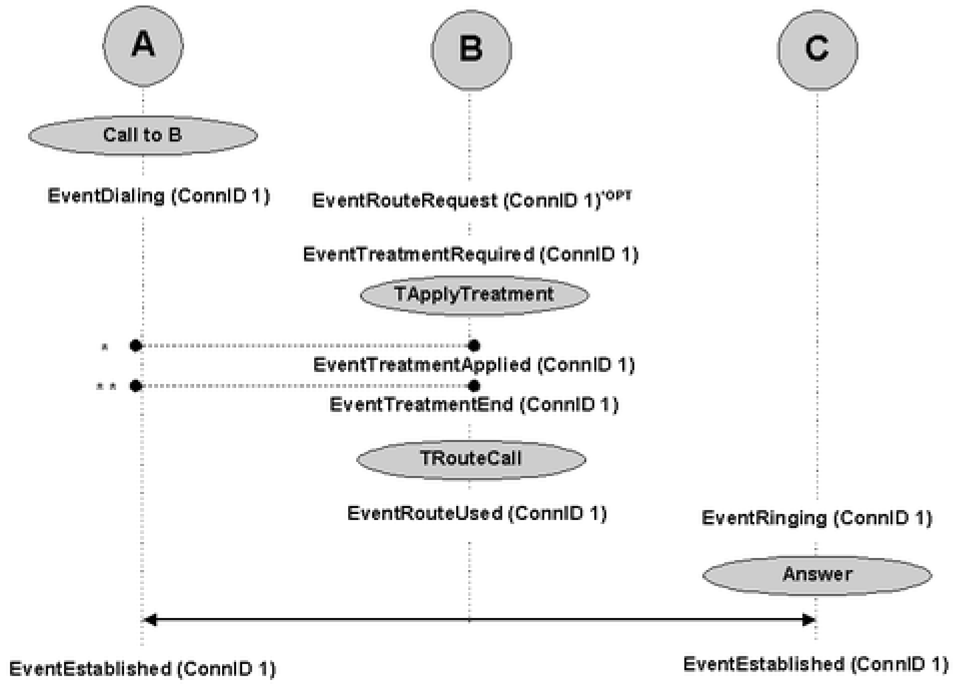
PARTY A	PARTY B	PARTY C
Make Call to B (TMakeCall) **or Inbound Call**		
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** *DIAL OtherDNRole **Destination** *DIAL	**EventRinging** ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**	
	Coverage Path	
		EventRinging ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **Covered**
	Answer (TAnswerCall)	
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** OtherDNRole **Destination**	**EventEstablished** ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**	**EventReleased** ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**		
**	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	
***	**EventReleased** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**

Call Treatment with Routing

The following graphic and table describe call treatment with routing.


A
B
C
Call to B
EventDialing (ConnID 1)
EventRouteRequest (ConnID 1)*OPT
EventTreatmentRequired (ConnID 1)
TApplyTreatment
*
EventTreatmentApplied (ConnID 1)
* *
EventTreatmentEnd (ConnID 1)
TRouteCall
EventRouteUsed (ConnID 1)
EventRinging (ConnID 1)
Answer
EventEstablished (ConnID 1)
EventEstablished (ConnID 1)

Call Treatment with Routing

PARTY A	PARTY B **(Routing Point)**	PARTY C
Call to B		
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **B** OtherDNRole **Destination** CallState **OK**	**EventRouteRequest** *OPT ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** **EventTreatmentRequired** ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**	
	Treatment Instruction **(TApplyTreatment)**	

PARTY A	PARTY B (Routing Point)	PARTY C
	EventTreatmentApplied ConnID **1** ThisDN **B** ThisDNRole **Destination** TreatmentType	
	EventTreatmentEnd ConnID **1** ThisDN **B** ThisDNRole **Destination** TreatmentType UserData	
	Route (TRouteCall)	
	EventRouteUsed ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** ThirdPartyDN **C** *OPT ThirdPartyDNRole **Destination** *OPT	**EventRinging** ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**
		Answer (TAnswerCall)
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Destination** OtherDN **C** OtherDNRole **Origination** CallState **OK**		**EventEstablished** ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**	
* *	**EventReleased** ConnID **1** ThisDN **A** OtherDN **B** CallState **OK**	**EventAbandoned** ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Destination**	

Interruption Point	PARTY A	PARTY B	PARTY C
		OtherDN **A** OtherDNRole **Origination**	

Predictive Dialing

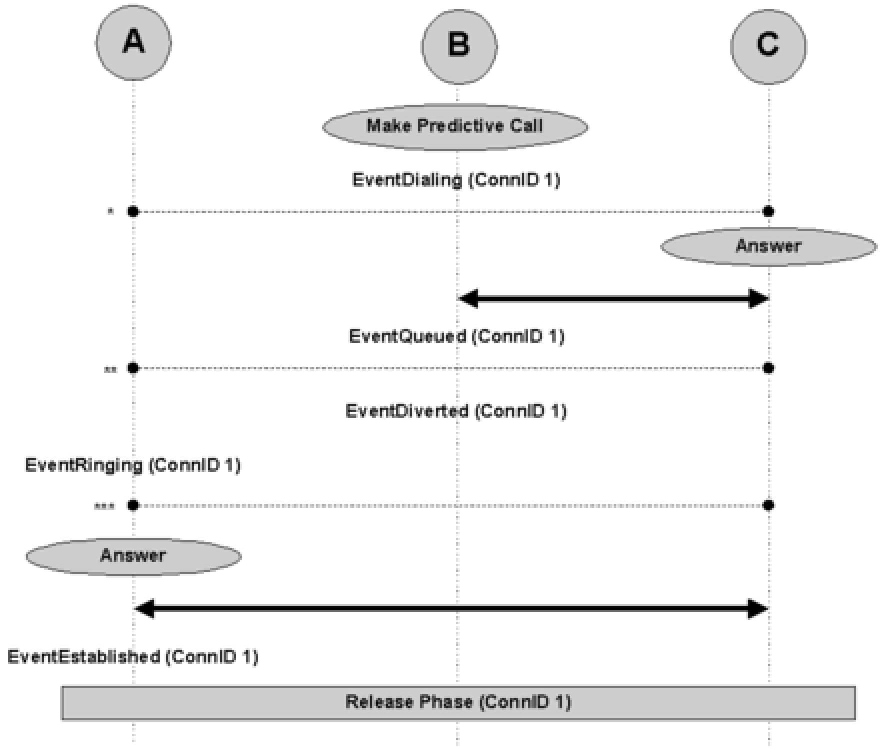
Note the following comments in the call models:

*OPT—Optional.

*DIAL—May be a dialed number or is not present if T-Server has no information about the other party.

Predictive Call

The following graphic and table describe a predictive call.



Predictive Call

PARTY A	PARTY B (ACD Group)	PARTY C
	Make Predictive Call **(TMakePredictiveCall)**	
	EventDialing	

PARTY A	PARTY B (ACD Group)	PARTY C
	ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Origination** OtherDN **C** *DIAL OtherDNRole **Destination**	
		Answer
	EventQueued ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Origination** CallState **OK** / **AnsweringMachineDetected** [a]	
	EventDiverted ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination** ThirdPartyDN **A** *OPT ThirdPartyDNRole **Origination** *OPT	
EventRinging ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination** CallState **OK**		
Answer (TAnswerCall)		
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination**		
	Release Phase (ConnID 1)	

a. If the switch reports that a call is connected to an answering machine, T-Server also attaches a key-value pair AnswerClass=AM to the call’s UserData.

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*		**EventReleased**	

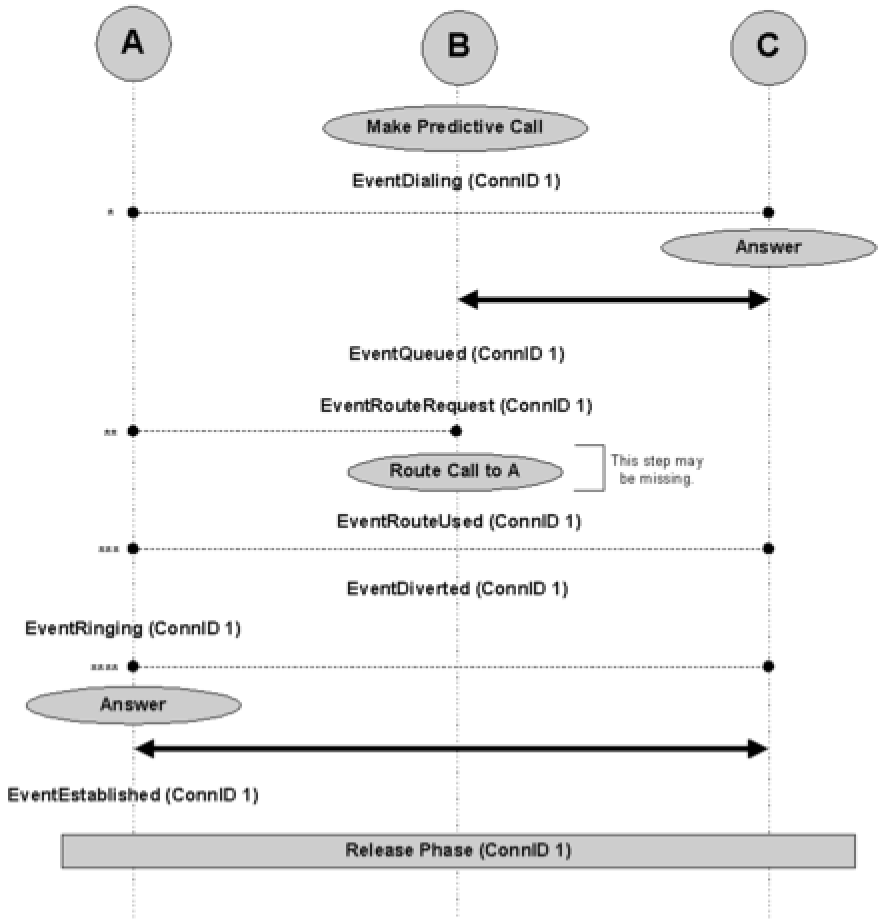
Interruption Point	PARTY A	PARTY B	PARTY C
		ConnID **1** ThisDN **B** OtherDN **C** CallState [a]	
**		**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **C** CallState **OK**	
***	**EventAbandoned** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		

a. CallState in this case may be any of the following:

- CallStateGeneralError
- CallStateSystemError
- CallStateBusy
- CallStateNoAnswer
- CallStateAnsweringMachineDetected
- CallStateFaxDetected
- CallStateAllTrunksBusy
- CallStateQueueFull
- CallStateDropped
- CallStateSitDetected
- CallStateSitInvalidnum
- CallStateSitVacant
- CallStateSitIntercept
- CallStateSitUnknown
- CallStateSitNocircuit
- CallStateSitReorder

Predictive Call with Routing

The following graphic and table describe a predictive call with routing.



Predictive Call with Routing

PARTY A	PARTY B (ACD Group)	PARTY C
	Make Predictive Call **(TMakePredictiveCall)**	
	EventDialing ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Origination** OtherDN **C** *DIAL OtherDNRole **Destination**	
		Answer
	EventQueued ConnID **1** ThisDN **B** ThisQueue **B**	

PARTY A	PARTY B (ACD Group)	PARTY C
	ThisDNRole **Origination** CallState **OK** / **FaxDetected** / **AnsweringMachineDetected** [a]	
	EventRouteRequest ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination**	
	Route Call to A **(TRouteCall)**	
	EventRouteUsed ConnID **1** ThisDN **B** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination** ThirdPartyDN **A** *OPT ThirdPartyDNRole **Origination** *OPT **EventDiverted** ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination** ThirdPartyDN **A** *OPT ThirdPartyDNRole **Origination** *OPT	
EventRinging ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination** CallState **OK**		
Answer (TAnswerCall)		
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Destination**		
	Release Phase (ConnID 1)	

a. If the switch reports that a call is connected to an answering machine, T-Server also attaches a key-value pair AnswerClass=AM to the call’s UserData.

Abnormal Call Flow

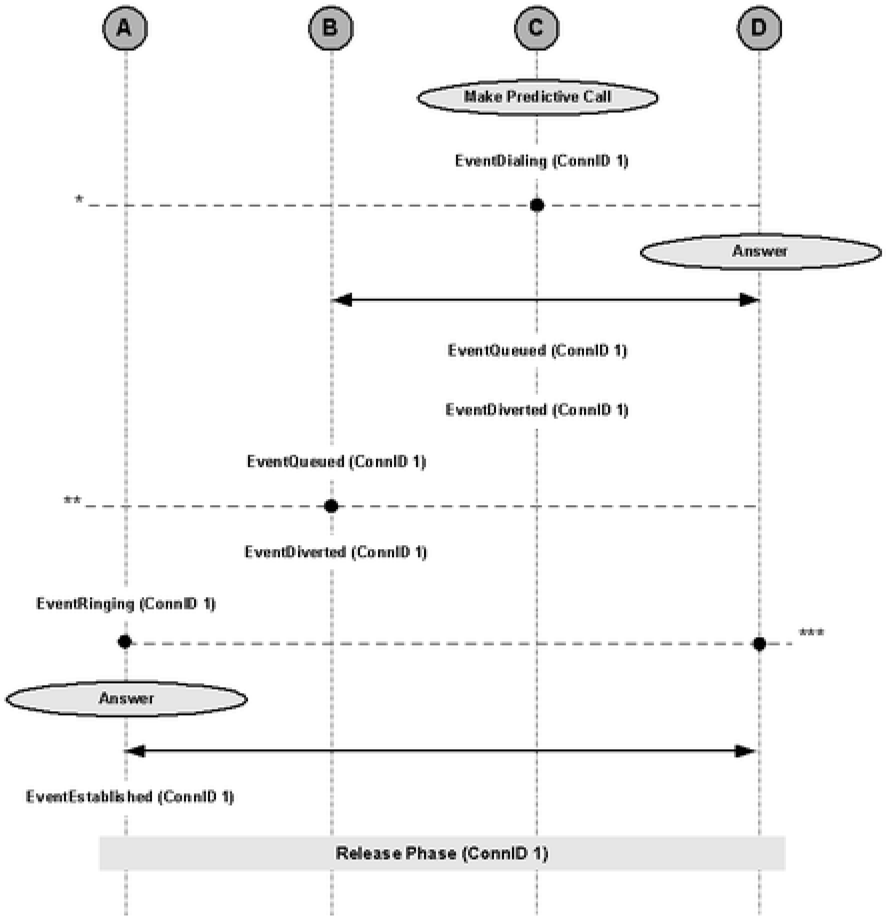
Interruption Point	PARTY A	PARTY B	PARTY C
*		**EventReleased** ConnID **1** ThisDN **B** OtherDN **C** CallState [a]	
** and ***		**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **C** CallState **OK**	
****	**EventAbandoned** ConnID **1** ThisDN **A** OtherDN **C** CallState **OK**		

a. CallState in this case may be any of the following:

- CallStateGeneralError
- CallStateSystemError
- CallStateBusy
- CallStateNoAnswer
- CallStateAnsweringMachineDetected
- CallStateFaxDetected
- CallStateAllTrunksBusy
- CallStateQueueFull
- CallStateDropped
- CallStateSitDetected
- CallStateSitInvalidnum
- CallStateSitVacant
- CallStateSitIntercept
- CallStateSitUnknown
- CallStateSitNocircuit
- CallStateSitReorder

Predictive Call (Connected to a Device Specified in Extensions)

The following graphic and table describe a predictive call (connected to a device specified in extensions).



Predictive Call (Connected to a Device Specified in Extensions)

PARTY A	**PARTY B (ACD Group** **Specified in the Extensions of TMakePredictiveCall)**	**PARTY C (Routing** **Point or ACD Group)**	**PARTY D**
		Make Predictive Call **(TMakePredictiveCall)**	
		EventDialing ConnID **1** ThisDN **C**	

PARTY A	PARTY B (ACD Group Specified in the Extensions of TMakePredictiveCall)	PARTY C (Routing Point or ACD Group)	PARTY D
		ThisQueue **C** ThisDNRole **Origination** OtherDN **D** *DIAL OtherDNRole **Destination**	
			Answer
		EventQueued ConnID **1** ThisDN **C** ThisQueue **C** ThisDNRole **Origination** CallState **OK**/**AnsweringMachine-Detected**	
		EventDiverted ConnID **1** ThisDN **C** ThisQueue **C** ThisDNRole **Origination** OtherDN **D** OtherDNRole **Destination** ThirdPartyDN **B** ThirdPartyDNRole **Origination**	
	EventQueued ConnID **1** This DN **B** ThisQueue **B** ThisDNRole **Origination** OtherDN **D** OtherDNRole **Destination**		
	EventDiverted ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Origination** OtherDN **D** OtherDNRole **Destination** ThirdPartyDN **A** *OPT ThirdPartyDNRole **Origination** *OPT		
EventRinging ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **D** OtherDNRole **Destination** CallState **OK**			
Answer			

PARTY A	PARTY B (ACD Group Specified in the Extensions of TMakePredictiveCall)	PARTY C (Routing Point or ACD Group)	PARTY D
(TAnswerCall)			
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN **D** OtherDNRole **Destination**			
Release Phase (ConnID 1)			

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C	PARTY D
*			**EventReleased** ConnID **1** ThisDN **C** OtherDN **D** CallState [a]	
**		**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **D** CallState **OK**		
***	**EventAbandoned** ConnID **1** ThisDN **A** OtherDN **D** CallState **OK**			

a. CallState in this case may be any of the following:

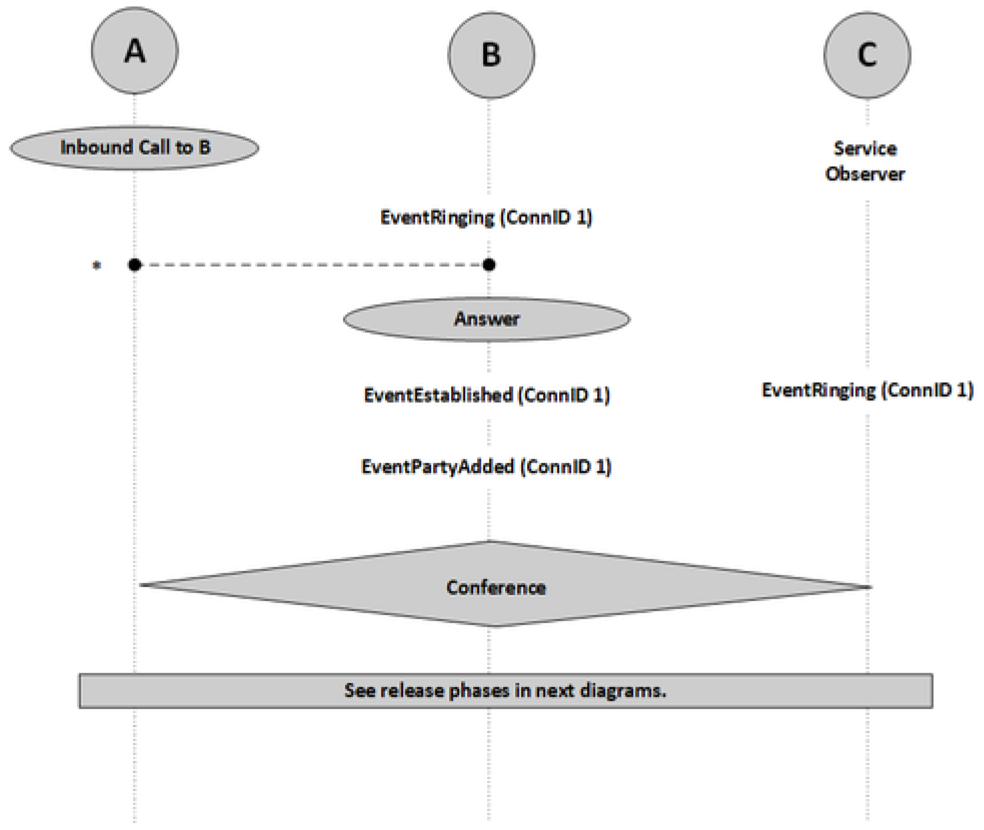
- CallStateGeneralError
- CallStateSystemError
- CallStateBusy
- CallStateNoAnswer
- CallStateAnsweringMachineDetected
- CallStateFaxDetected

- CallStateAllTrunksBusy
- CallStateQueueFull
- CallStateDropped
- CallStateSitDetected
- CallStateSitInvalidnum
- CallStateSitVacant
- CallStateSitIntercept
- CallStateSitUnknown
- CallStateSitNocircuit
- CallStateSitReorder

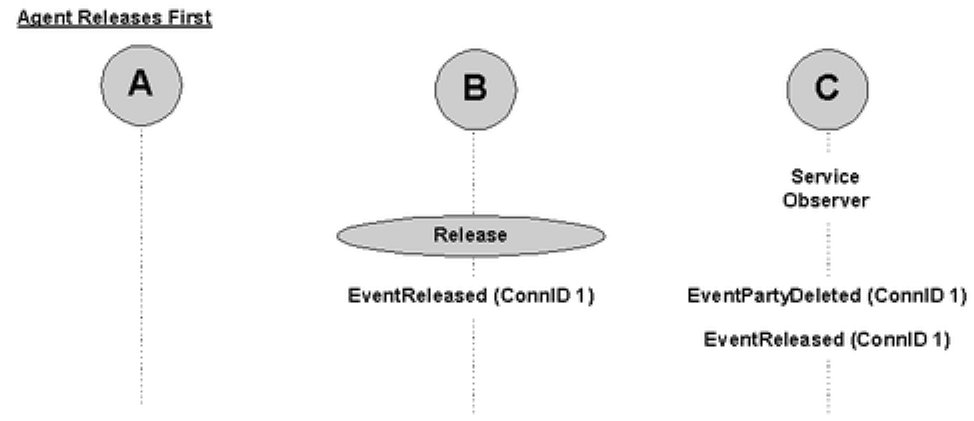
Monitoring Calls

Service Observing on Agent

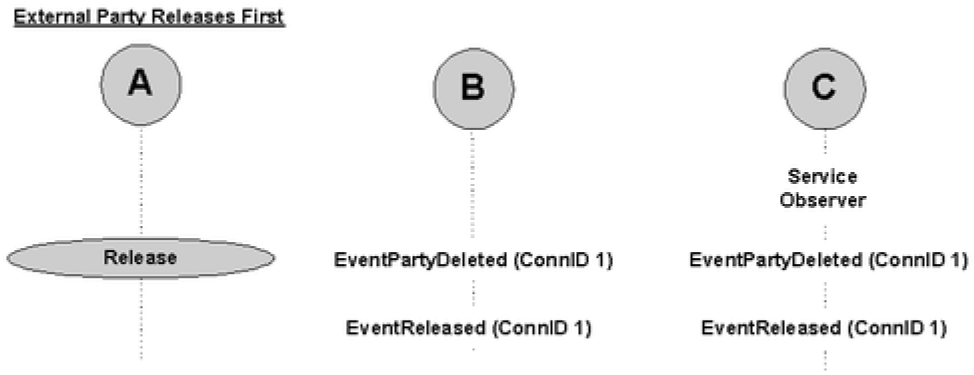
The following graphics and table describe service observing on an agent.



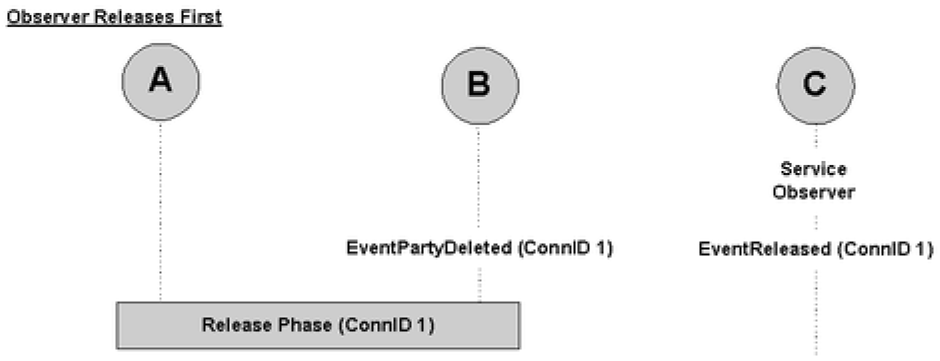
Service Observing on Agent



Agent Releases First


External Party Releases First
A
B
C
Service Observer
Release
EventPartyDeleted (ConnID 1)
EventReleased (ConnID 1)
EventPartyDeleted (ConnID 1)
EventReleased (ConnID 1)

External Party Releases First



Observer Releases First

PARTY A **(External)**	**PARTY B**	**PARTY C** **(Service Observer)**
Inbound Call		
	EventRinging ConnID **1**	

PARTY A **(External)**	**PARTY B**	**PARTY C** **(Service Observer)**
	ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**	
	Answer (TAnswerCall)	
	EventEstablished ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **OK**	**EventRinging** ConnID **1** ThisDN **C** ThisDNRole **Observer** OtherDN **A** [a] OtherDNRole **Origination** [b] CallState **Bridged**
	EventPartyAdded ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **C** OtherDNRole **Observer** CallState **Bridged**	**EventEstablished** ConnID **1** ThisDN **C** ThisDNRole **Observer** OtherDN **A** OtherDNRole **Origination** CallState **Bridged**
	Conference	
	Release Phase, see description below.	

a. T-Servers for some switches use the party that initialized the Service Observer instead of Party A.
b. T-Servers for some switches use the role of the party that initialized the Service Observer instead of the role of Party A.

Agent Releases First

PARTY A **(External)**	**PARTY B**	**PARTY C** **(Service Observer)**
	Release (TReleaseCall)	
	EventReleased ConnID **1** ThisDN **B** ThisDNRole **Destination** CallState **OK**	**EventPartyDeleted** ConnID **1** ThisDN **C** ThisDNRole **Observer** OtherDN **B** OtherDNRole **DeletedParty** CallState **OK**
		EventReleased

PARTY A **(External)**	**PARTY B**	**PARTY C** **(Service Observer)**
		ConnID **1** ThisDN **C** ThisDNRole **Observer** OtherDN **A** CallState **OK**

External Party Releases First

PARTY A **(External)**	**PARTY B**	**PARTY C** **(Service Observer)**
External Party Releases a Call		
	EventPartyDeleted ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **DeletedParty** ThirdPartyDNRole **Observer** [a] ThirdPartyDN **C** [a] CallState **OK**	**EventPartyDeleted** ConnID **1** ThisDN **C** ThisDNRole **Observer** OtherDN **A** OtherDNRole **DeletedParty** CallState **OK**
	EventReleased ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **C** CallState **OK**	**EventReleased** ConnID **1** ThisDN **C** ThisDNRole **Observer** OtherDN **B** CallState **OK**

a. The attribute contains observer information.

Observer Releases First

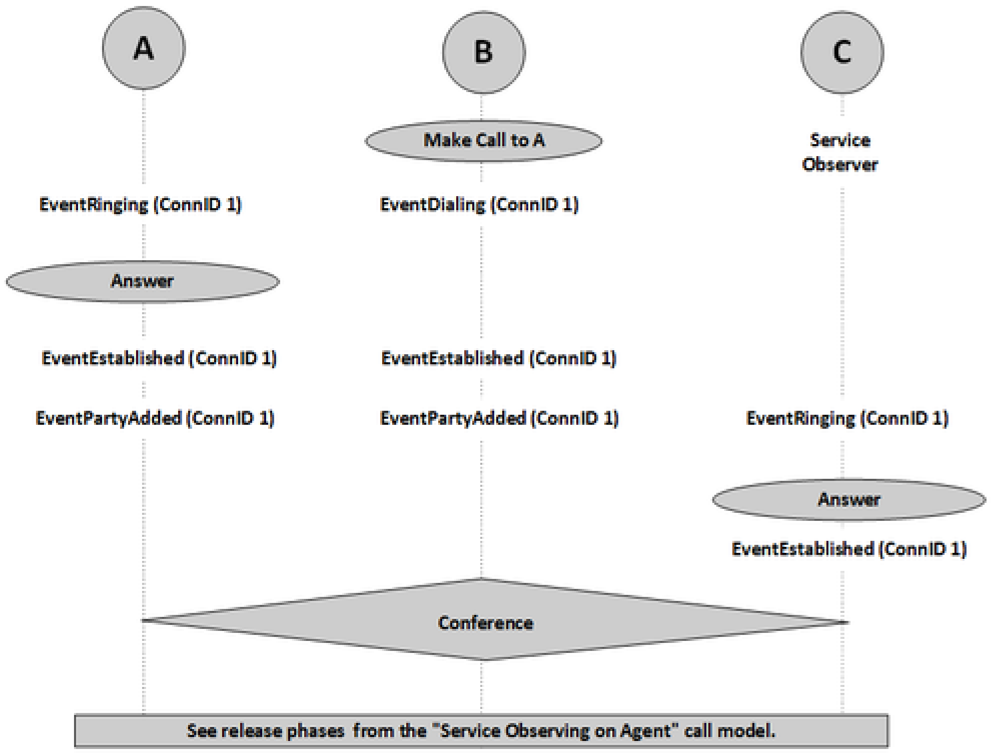
PARTY A **(External)**	**PARTY B**	**PARTY C** **(Service Observer)**
		Observer Releases a Call
	EventPartyDeleted ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **C** OtherDNRole **Observer** ThirdPartyDNRole **DeletedBy** ThirdPartyDN **C** CallState **OK**	**EventReleased** ConnID **1** ThisDN **C** ThisDNRole **Observer** CallState **OK**
Release Phase (ConnID 1)		

Abnormal Call Flow

Interruption Point	PARTY A	PARTY B	PARTY C
*		**EventAbandoned** ConnID **1** ThisDN **B** OtherDN **A** CallState **OK**	

Service Observing for Agent-Initiated Call

The following graphic and table describe service observing for an agent-initiated call.


A
B
C
Make Call to A
Service Observer
EventRinging (ConnID 1)
EventDialing (ConnID 1)
Answer
EventEstablished (ConnID 1)
EventEstablished (ConnID 1)
EventPartyAdded (ConnID 1)
EventPartyAdded (ConnID 1)
EventRinging (ConnID 1)
Answer
EventEstablished (ConnID 1)
Conference
See release phases from the "Service Observing on Agent" call model.

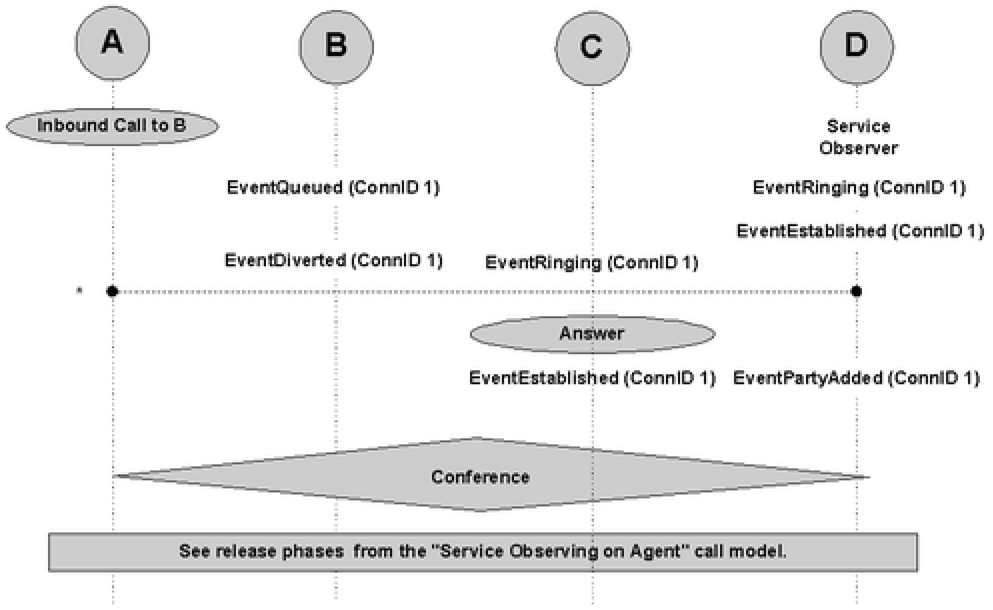
Service Observing for Agent-Initiated Call

PARTY A	PARTY B	PARTY C
EventRinging ConnID **1** ThisDN **A** ThisDNRole **Destination** OtherDN **B** OtherDNRole **Origination** CallState **OK**	**EventDialing** ConnID **1** ThisDN **B** ThisDNRole **Origination** OtherDN **A** OtherDNRole **Destination** CallState **OK**	

PARTY A	PARTY B	PARTY C
Answer		
EventEstablished ConnID **1** ThisDN **A** ThisDNRole **Destination** OtherDN **B** OtherDNRole **Origination** CallState **OK**	**EventEstablished** ConnID **1** ThisDN **B** ThisDNRole **Origination** OtherDN **A** OtherDNRole **Destination** CallState **OK**	
EventPartyAdded ConnID **1** ThisDN **A** ThisDNRole **Destination** OtherDN **C** OtherDNRole **Observer** CallState **Bridged**	**EventPartyAdded** ConnID **1** ThisDN **B** ThisDNRole **Origination** OtherDN **C** OtherDNRole **Observer** CallState **Bridged**	**EventRinging** ConnID **1** ThisDN **C** ThisDNRole **Observer** OtherDN **B** OtherDNRole **Origination** CallState **Bridged**
		Answer
		EventEstablished ConnID **1** ThisDN **C** ThisDNRole **Observer** CallState **Bridged**
Conference		
Release Phase, see description in Service Observer on Agent.		

Service Observing on Queue

The following graphic and table describe service observing on the queue.


A
B
C
D
Inbound Call to B
Service Observer
EventQueued (ConnID 1)
EventRinging (ConnID 1)
EventEstablished (ConnID 1)
EventDiverted (ConnID 1)
EventRinging (ConnID 1)
Answer
EventEstablished (ConnID 1)
EventPartyAdded (ConnID 1)
Conference
See release phases from the "Service Observing on Agent" call model.

Service Observing on Queue

PARTY A **(External)**	**PARTY B**	**PARTY C**	**Party D (Observer)**
Inbound Call			
	EventQueued ConnID **1** ThisDN **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination**		**EventRinging** ConnID **1** ThisDN **D** ThisDNRole **Observer** OtherDN **A** OtherDNRole **Origination** CallState **Bridged**
			EventEstablished ConnID **1** ThisDN **D** ThisDNRole **Observer** OtherDN **A** OtherDNRole **Origination** CallState **Bridged**
	EventDiverted ConnID **1** ThisDN **B** ThisQueue **B** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** ThirdPartyDN **C** *OPT ThirdPartyDNRole **Destination** *OPT	**EventRinging** ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **Bridged**	
		Answer	

PARTY A **(External)**	**PARTY B**	**PARTY C**	**Party D** **(Observer)**
		(TAnswerCall)	
		EventEstablished ConnID **1** ThisDN **C** ThisDNRole **Destination** OtherDN **A** OtherDNRole **Origination** CallState **Bridged** Extensions: OrigDN-1=A OrigDN-2=D	**EventPartyAdded** ConnID **1** ThisDN **D** ThisDNRole **Observer** OtherDN **C** OtherDNRole **NewParty** ThirdPartyDN **C** ThirdPartyDNRole **AddedBy** CallState **Bridged**
Conference			
Release Phase, see description in Service Observer on Agent.			

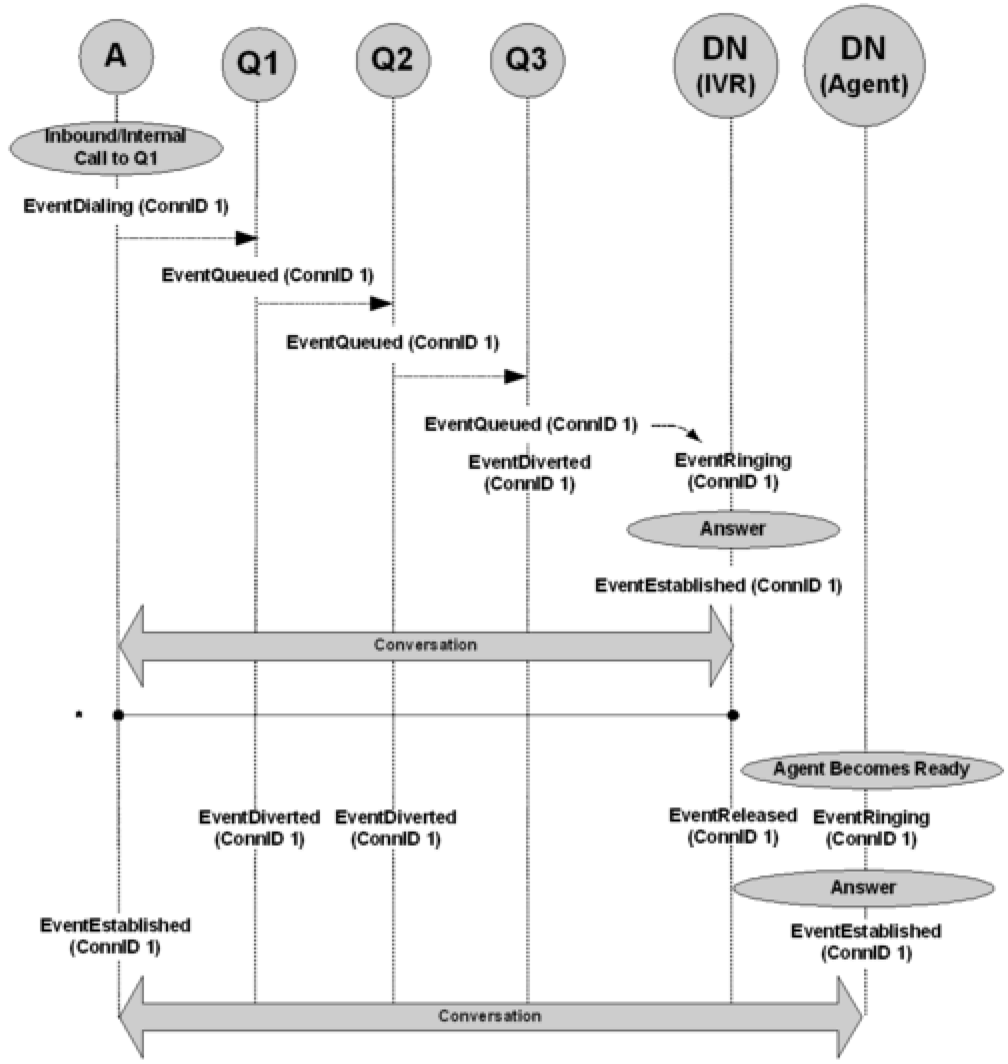
Abnormal Call Flow

Interruption Point	**PARTY A**	**PARTY B**	**PARTY C**	**PARTY D**
*			**EventAbandoned** ConnID **1** ThisDN **C** OtherDN **A** CallState **OK**	**EventReleased** ConnID **1** ThisDN **D** ThisDNRole **Observer** OtherDN **A** CallState **OK**

Working With Queues

Multiple-Queue Call Treated at an IVR Port: Treatment at IVR Queue

The following graphic and table describe a multiple-queue call treated an an IVR port: treatment at the IVR queue.



Multiple Queue, Call Treated at an IVR Port: Treatment at IVR Queue

A	Q1	Q2	Q3	IVR	Agent
Inbound **/Internal Call to Q1**	**Call to Q1**				
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN* **Q1** OtherDNRole **Destination**					
	EventQueued ConnID **1** ThisDN **Q1** ThisQueue **Q1** OtherDN **A**				
		Call Placed in Second Queue			
		EventQueued ConnID **1** ThisDN **Q2** ThisQueue **Q2** OtherDN **A**			
			Call Placed in IVR Queue for Treatment When No Agents Ready		
			EventQueued ConnID **1** ThisDN **Q2** ThisQueue **Q2** OtherDN **A**		
			EventDiverted ConnID **1** ThisDN **Q3** ThisQueue **Q3** OtherDN **A** ThirdPartyDN **IVR DN** CallState **ConverseOn**	**EventRinging** ConnID **1** ThisDN **IVR** ThisQueue **Q3** OtherDN **A** CallState **ConverseOn**	
				Answer	
				EventEstablished	

A	Q1	Q2	Q3	IVR	Agent
				ConnID **1** ThisDN **IVR** ThisQueue **Q3** OtherDN **A**	
					Agent Ready
	EventDiverted ConnID **1** ThisDN **RQ2** ThisQueue **RQ2** OtherDN **A** ThirdPartyDN **AgentDN**	**EventDiverted** ConnID **1** ThisDN **Q2** ThisQueue **Q2** OtherDN **A** ThirdPartyDN **AgentDN**		**EventReleased**[a] ConnID **1** ThisDN **IVR** ThisQueue **Q3** OtherDN **A**	**EventRinging** ConnID **1** ThisDN **AgentDN** ThisQueue **Q1** OtherDN **A**
					Answer
EventEstablished[b] ConnID **1** ThisDN **A** OtherDN **AgentDN** CallState **OK**					**EventEstablished** ConnID **1** ThisDN **AgentDN** ThisQueue **Q1** OtherDN **A** CallState **OK**

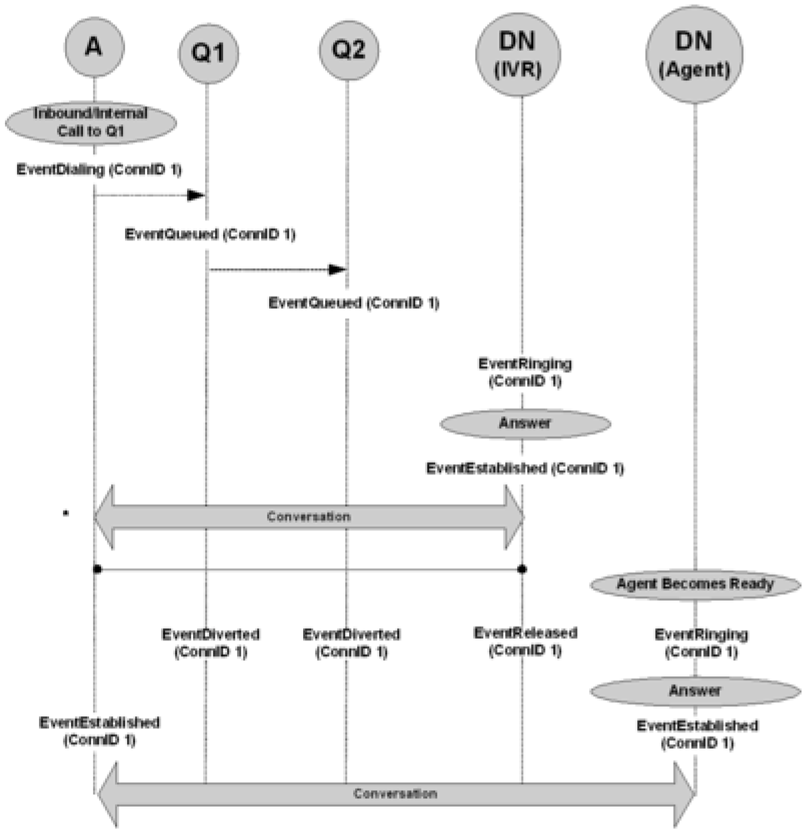
a. `EventReleased` can occur before an agent becomes available because the IVR finishes call treatment. b. In some deployments, `EventEstablished` for party A can occur at the same time as the IVR `EventEstablished`, especially if a call comes through the PSTN.

Abnormal Call Flow

Interruption Point	A	Q1	Q2	Q3	IVR	Agent
*	**EventReleased** OtherDN **Q1**	**EventAbandoned** ConnID **1** ThisDN **Q1** OtherDN **A**	**EventAbandoned** ConnID **1** ThisDN **Q2** OtherDN **A**	**EventAbandoned** ConnID **1** ThisDN **Q3** OtherDN **A**	**EventReleased** ConnID **1** ThisDN **IVR** OtherDN **A**	

Multiple-Queue, Call Treated at an IVR Port: Direct Treatment at IVR Port

The following graphic and table describe a multiple-queue call treated at an IVR port: direct treatment at the IVR queue.



Multiple Queue, Call Treated at an IVR Port: Direct Treatment at IVR Port

External Party	Q1	Q2	IVR	Agent
Inbound **/Internal Call to Q1**	**Call to Q1**			
EventDialing ConnID **1** ThisDN **A** ThisDNRole **Origination** OtherDN* **Q1** OtherDNRole **Destination**				
	EventQueued ConnID **1** ThisDN **Q1** ThisQueue **Q1** OtherDN **A**			
		Call Placed in Second Queue		

External Party	Q1	Q2	IVR	Agent
		EventQueued ConnID **1** ThisDN **Q2** ThisQueue **Q2** OtherDN **A**		
			Call Placed Directly to IVR Port	
			EventRinging ConnID **1** ThisDN **IVR** OtherDN **A** CallState **ConverseOn**	
			Answer	
			EventEstablished ConnID **1** ThisDN **IVR** OtherDN **A**	
				Agent Ready
	EventDiverted ConnID **1** ThisDN **RQ2** ThisQueue **RQ2** OtherDN **A** ThirdPartyDN **AgentDN**	**EventDiverted** ConnID **1** ThisDN **Q2** ThisQueue **Q2** OtherDN **A** ThirdPartyDN **AgentDN**	**EventReleased** [a] ConnID **1** ThisDN **IVR** OtherDN **A**	**EventRinging** ConnID **1** ThisDN **AgentDN** ThisQueue **Q1** OtherDN **A**
				Answer
EventEstablished [b] ConnID **1** ThisDN **A** OtherDN **AgentDN** CallState **OK**				**EventEstablished** ConnID **1** ThisDN **AgentDN** ThisQueue **Q1** OtherDN **A** CallState **OK**

a. `EventReleased` can occur before an agent becomes available because the IVR finishes call treatment. b. In some deployments, `EventEstablished` for party A can occur at the same time as the IVR `EventEstablished`, especially if a call comes through the PSTN.

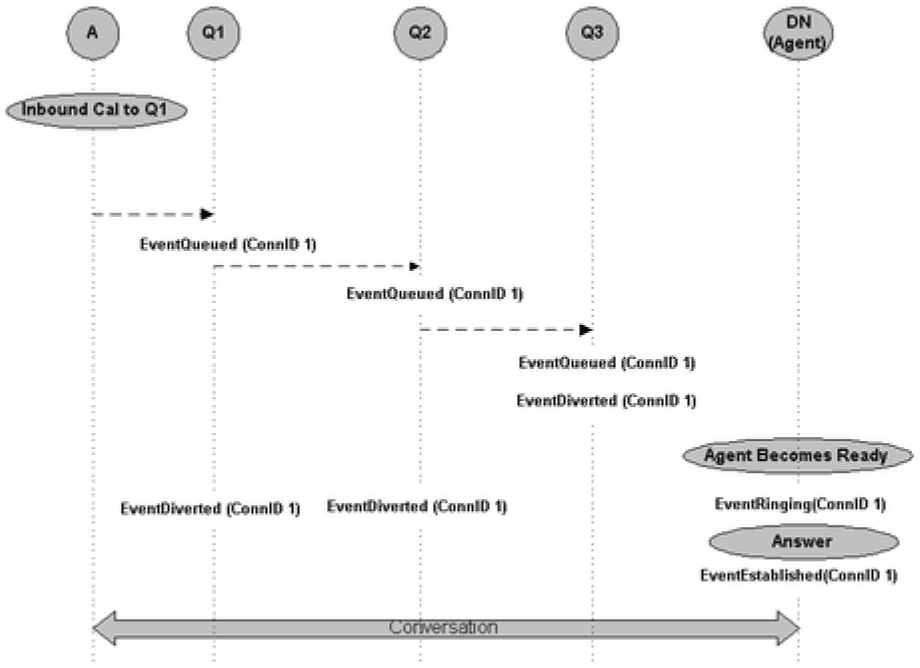
Abnormal Call Flow

Interruption Point	External Party	Q1	Q2	IVR	Agent
*	**EventReleased** OtherDN **Q1**	**EventAbandoned** ConnID **1** ThisDN **Q1**	**EventAbandoned** ConnID **1** ThisDN **Q2**	**EventReleased** ConnID **1** ThisDN **IVR**	

Interruption Point	External Party	Q1	Q2	IVR	Agent
		OtherDN **A**	OtherDN **A**	OtherDN **A**	

Multiple-Queue Call: Call Removed from Queue

The following graphic and table describe a multiple-queue call: with the call removed from the queue.



Multiple-Queue Call: Call Removed from Queue

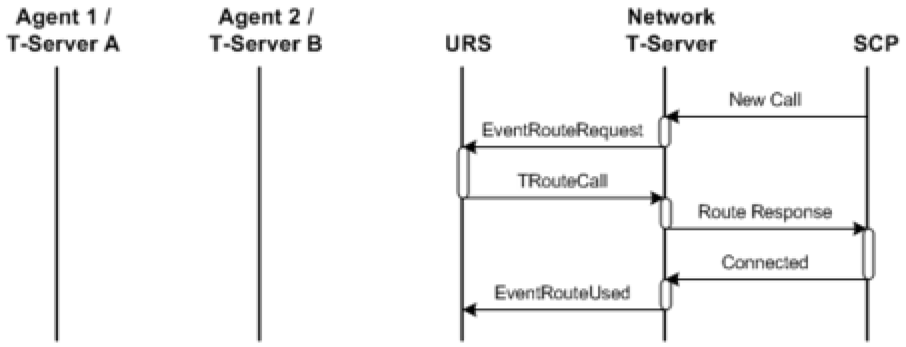
A	Q1	Q2	IVR	Agent
Inbound Call to Q1	**Call to Q1**			
	EventQueued ConnID **1** ThisDN **Q1** ThisQueue **Q1** OtherDN **A**			
		Call Placed in		

A	Q1	Q2	IVR	Agent
		Second Queue		
		EventQueued ConnID **1** ThisDN **Q2** ThisQueue **Q2** OtherDN **A**		
			Call Placed in **Third Queue for Treatment When No Agents Ready**	
			EventQueued ConnID **1** ThisDN **Q3** ThisQueue **Q3** OtherDN **A**	
			Call Cleared **from Third Queue**	
			EventDiverted ConnID **1** ThisDN **Q3** ThisQueue **Q3** OtherDN **A** CallState **Cleared**	
				Agent Ready
	EventDiverted ConnID **1** ThisDN **Q1** ThisQueue **Q1** OtherDN **A** ThirdPartyDN **AgentDN**	**EventDiverted** ConnID **1** ThisDN **Q2** ThisQueue **Q2** OtherDN **A** ThirdPartyDN **AgentDN**		**EventRinging** ConnID **1** ThisDN **AgentDN** ThisQueue **Q1** OtherDN **A** CallState **OK**
				Answer
				EventEstablished ConnID **1** ThisDN **AgentDN** ThisQueue **Q1** OtherDN **A** CallState **OK**

Network T-Server Attended Transfer Call Flows

Standard Network Call Initiation

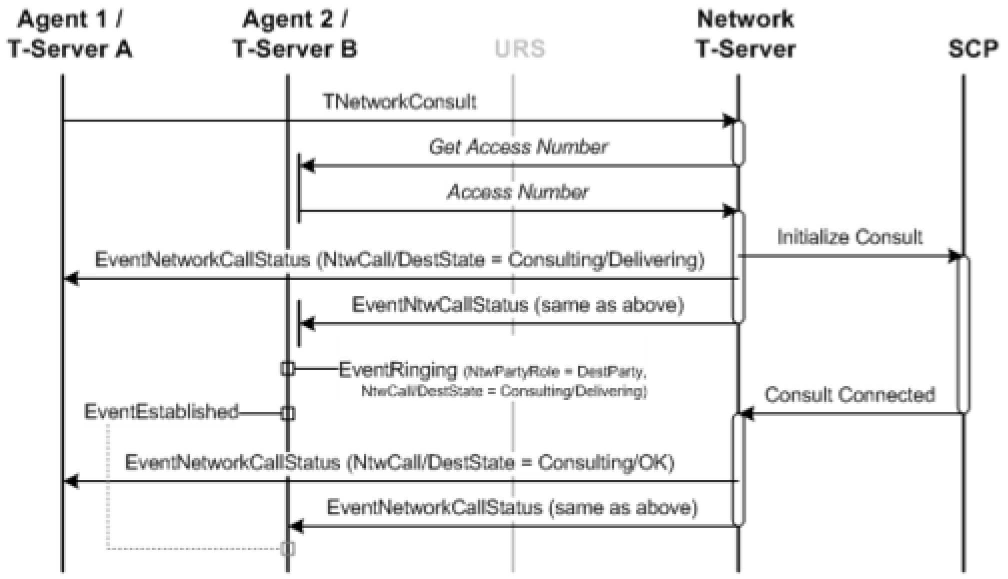
The Standard Network Call Initiation figure illustrates the standard call setup for a call entering a Genesys environment through a Network T-Server. The diagrams that follow it in this section assume the completion of this stage and present common network attended transfer scenarios and a few error cases.



Standard Network Call Initiation

Consultation Leg Initiation, Specific Destination

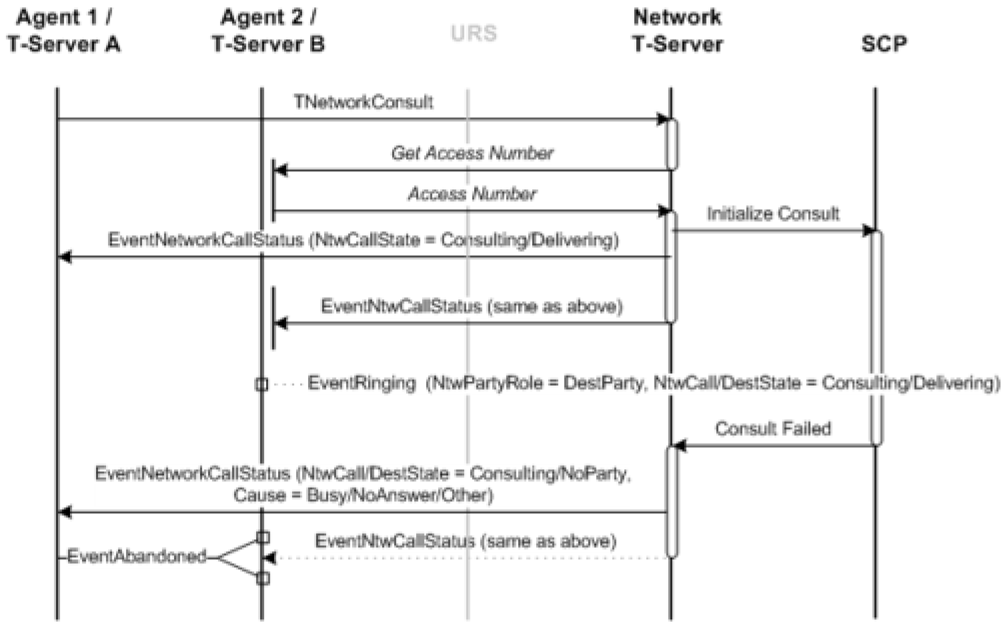
The following figure illustrates the creation of the consultation leg in a case where the destination DN and its location are provided in the `TNetworkConsult()` function call.


Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
TNetworkConsult
Get Access Number
Access Number
Initialize Consult
EventNetworkCallStatus (NtwCall/DestState = Consulting/Delivering)
EventNtwCallStatus (same as above)
EventRinging (NtwPartyRole = DestParty, NtwCall/DestState = Consulting/Delivering)
Consult Connected
EventEstablished
EventNetworkCallStatus (NtwCall/DestState = Consulting/OK)
EventNetworkCallStatus (same as above)

Consultation Leg Initiation, Specific Destination

Failed Consultation: Specific Target

In the following figure, the request to consult has failed. Events shown with dotted lines are distributed only when the call is delivered to the intended destination, but not answered (for instance, when `State = NoAnswer`). Since the consult request in this case is made to a specific DN and location, an explicit reconnect request (or another `TNetworkConsult` request, if allowed by the given SCP) from Agent 1 is required. Until such a request is made, the call remains in the `Consulting/NoParty` state.

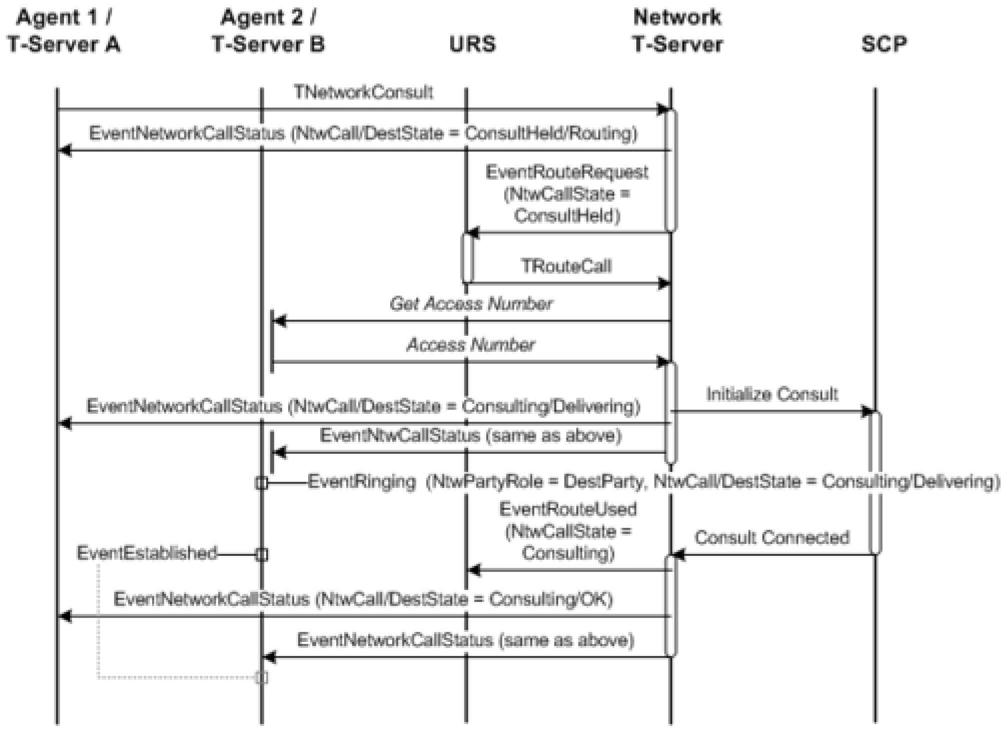

Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
TNetworkConsult
Get Access Number
Access Number
Initialize Consult
EventNetworkCallStatus (NtwCallState = Consulting/Delivering)
EventNtwCallStatus (same as above)
EventRinging (NtwPartyRole = DestParty, NtwCall/DestState = Consulting/Delivering)
Consult Failed
EventNetworkCallStatus (NtwCall/DestState = Consulting/NoParty, Cause = Busy/NoAnswer/Other)
EventNtwCallStatus (same as above)
EventAbandoned

Failed Consultation: Direct Target

Consultation Leg Initiation, URS Selected Destination

The following figure illustrates the creation of the consultation leg in a case where the destination DN and its location are provided by URS. In this case the original caller is still connected to the call (speaking with Agent 1) while URS selects the target. The caller is then placed on hold the moment the consultation leg is created. Depending on the given SCP's capabilities, other scenarios are also possible. For example:

- The original caller might be put on hold immediately (as is the case with conventional local two-step transfers). In such a case, the value for `NetworkState` for the first event is `Consulting/Routing`.
- The original caller might be connected to the call throughout the entire consult and delivery phase, and then put on hold only when the destination (Agent 2) answers the call. In this case, the second event has a value of `ConsultHeld` instead of `Consulting`.

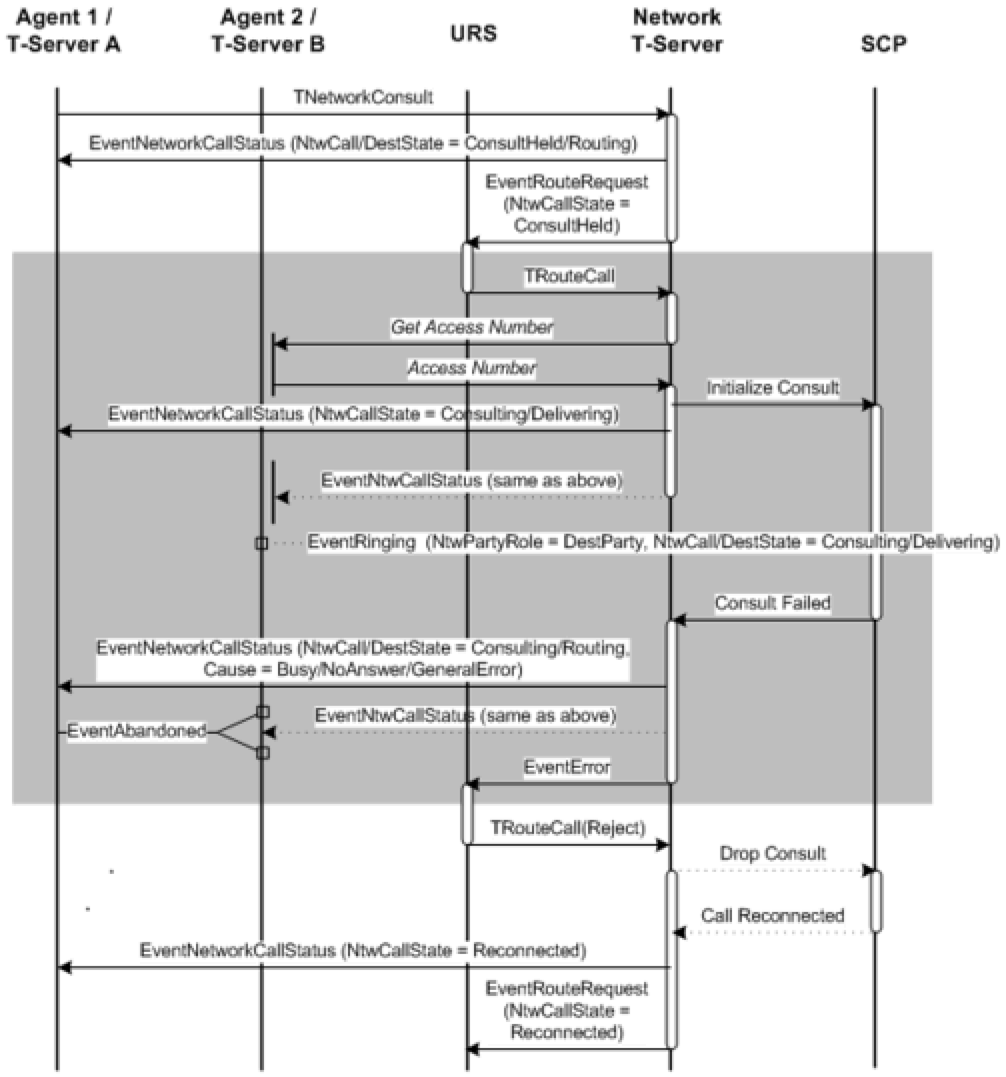

Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
TNetworkConsult
EventNetworkCallStatus (NtwCall/DestState = ConsultHeld/Routing)
EventRouteRequest (NtwCallState = ConsultHeld)
TRouteCall
Get Access Number
Access Number
EventNetworkCallStatus (NtwCall/DestState = Consulting/Delivering)
Initialize Consult
EventNtwCallStatus (same as above)
EventRinging (NtwPartyRole = DestParty, NtwCall/DestState = Consulting/Delivering)
EventRouteUsed (NtwCallState = Consulting)
Consult Connected
EventEstablished
EventNetworkCallStatus (NtwCall/DestState = Consulting/OK)
EventNetworkCallStatus (same as above)

Consultation Leg Initiation, URS Selected Destination

Failed Consultation: URS Selected Destination

The following figure shows a failed request for consultation. Events shown with dotted lines are distributed only when the call is delivered to the intended destination, but not answered (for instance, when `State = NoAnswer`). Because URS controls consultation initiation in this case, no explicit reconnection is required. URS continues to offer different route targets until the consultation is connected. (The shaded portion of the diagram may repeat several times.) At this point:

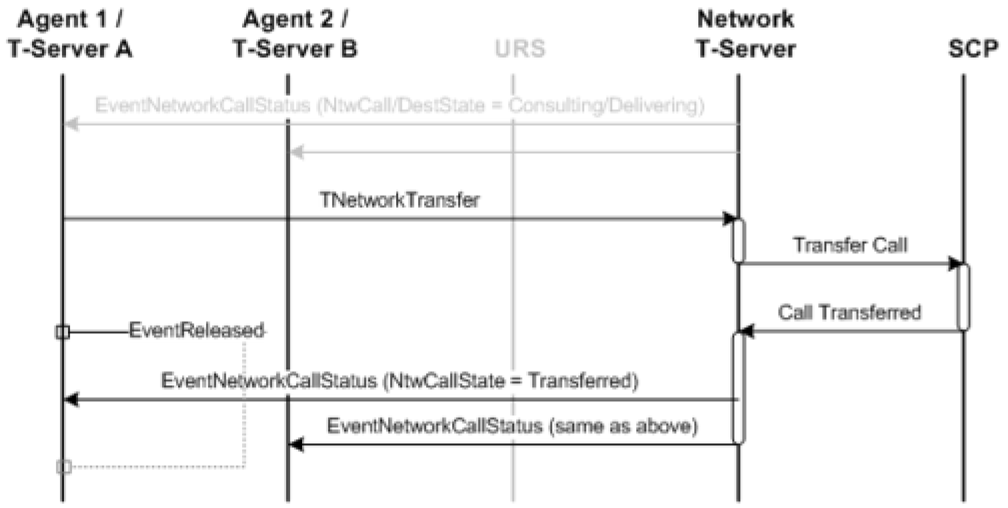
- Reconnect can still occur manually.
- URS can reconnect using a reject or default route instruction.


Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
TNetworkConsult
EventNetworkCallStatus (NtwCall/DestState = ConsultHeld/Routing)
EventRouteRequest (NtwCallState = ConsultHeld)
TRouteCall
Get Access Number
Access Number
Initialize Consult
EventNetworkCallStatus (NtwCallState = Consulting/Delivering)
EventNtwCallStatus (same as above)
EventRinging (NtwPartyRole = DestParty, NtwCall/DestState = Consulting/Delivering)
Consult Failed
EventNetworkCallStatus (NtwCall/DestState = Consulting/Routing, Cause = Busy/NoAnswer/GeneralError)
EventNtwCallStatus (same as above)
EventAbandoned
EventError
TRouteCall(Reject)
Drop Consult
Call Reconnected
EventNetworkCallStatus (NtwCallState = Reconnected)
EventRouteRequest (NtwCallState = Reconnected)

Failed Consultation: URS Selected Target

Transfer/Conference Completion: Explicit

The following figure illustrates the completion of either a network-attended transfer or conference. It assumes that the call had a successful consultation phase for its transfer or conference completion to be valid. This assumption is necessary since `EventNetworkCallStatus` is a direct response to the `TNetworkTransfer` feature request. The Network T-Server sends Agent 1 `EventNetworkCallStatus` regardless of whether the call has already been released.


Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
EventNetworkCallStatus (NtwCall/DestState = Consulting/Delivering)
TNetworkTransfer
Transfer Call
Call Transferred
EventReleased
EventNetworkCallStatus (NtwCallState = Transferred)
EventNetworkCallStatus (same as above)

Transfer/Conference Completion: Explicit

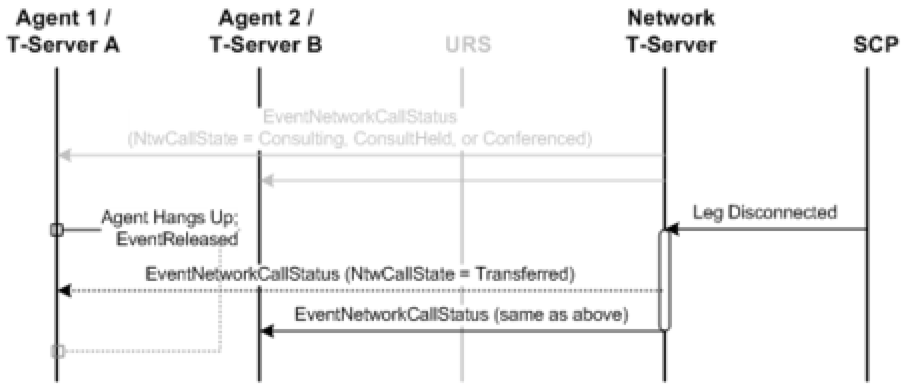
Transfer Completion: Implicit

An implicit transfer completion occurs when Agent 1's channel becomes disconnected and the call has a status of `Consulting`, `ConsultHeld`, or `Conferencing`.

Important

This scenario applies only when the disconnection is with respect to Agent 1. (For details on a disconnection with Agent 2, see Implicit Reconnection (by SCP) and Implicit Reconnection (by Network T-Server). For a disconnection with respect to the caller, see Caller Abandonment.)

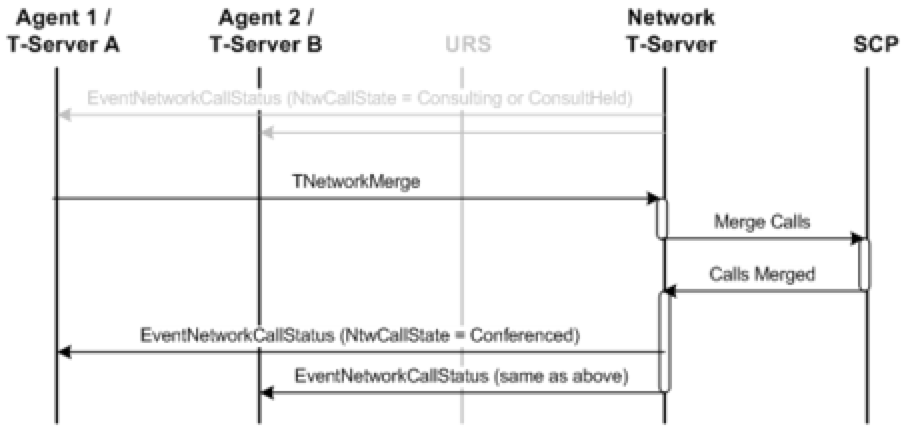
`EventNetworkCallStatus` is sent to Agent 1 only if T-Server A delays the release of `EventReleased` for some reason.


Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
EventNetworkCallStatus (NtwCallState = Consulting, ConsultHeld, or Conferenced)
Leg Disconnected
Agent Hangs Up; EventReleased
EventNetworkCallStatus (NtwCallState = Transferred)
EventNetworkCallStatus (same as above)

Network Attended Transfer Completion: Implicit

Conference Completion

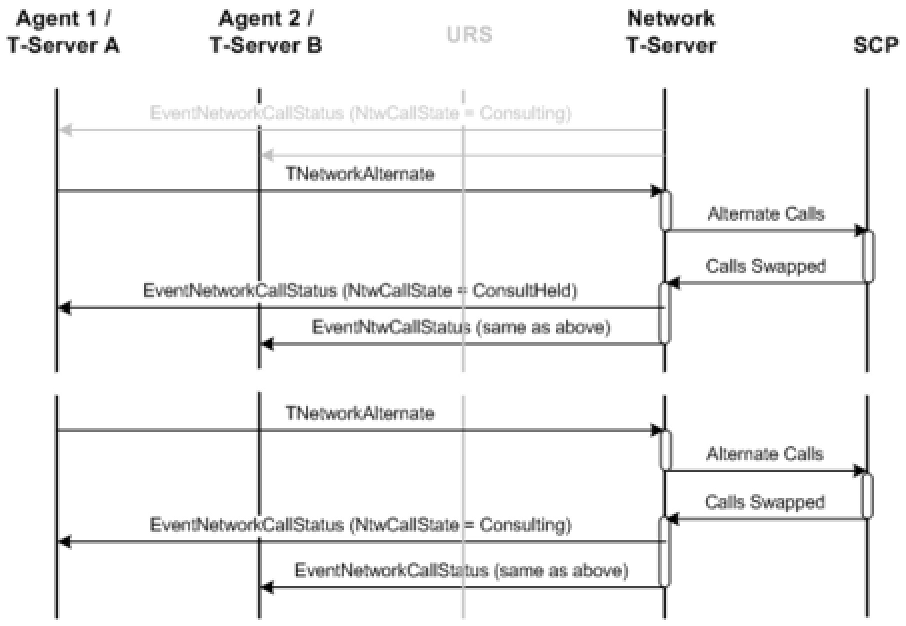
The following figure illustrates a standard conference completion.


Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
EventNetworkCallStatus (NtwCallState = Consulting or ConsultHeld)
TNetworkMerge
Merge Calls
Calls Merged
EventNetworkCallStatus (NtwCallState = Conferenced)
EventNetworkCallStatus (same as above)

Conference Completion

Alternate Call Service

The following figure contains two variations of the service to show different possible network states for the events.


Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
EventNetworkCallStatus (NtwCallState = Consulting)
TNetworkAlternate
Alternate Calls
Calls Swapped
EventNetworkCallStatus (NtwCallState = ConsultHeld)
EventNtwCallStatus (same as above)
TNetworkAlternate
Alternate Calls
Calls Swapped
EventNetworkCallStatus (NtwCallState = Consulting)
EventNetworkCallStatus (same as above)

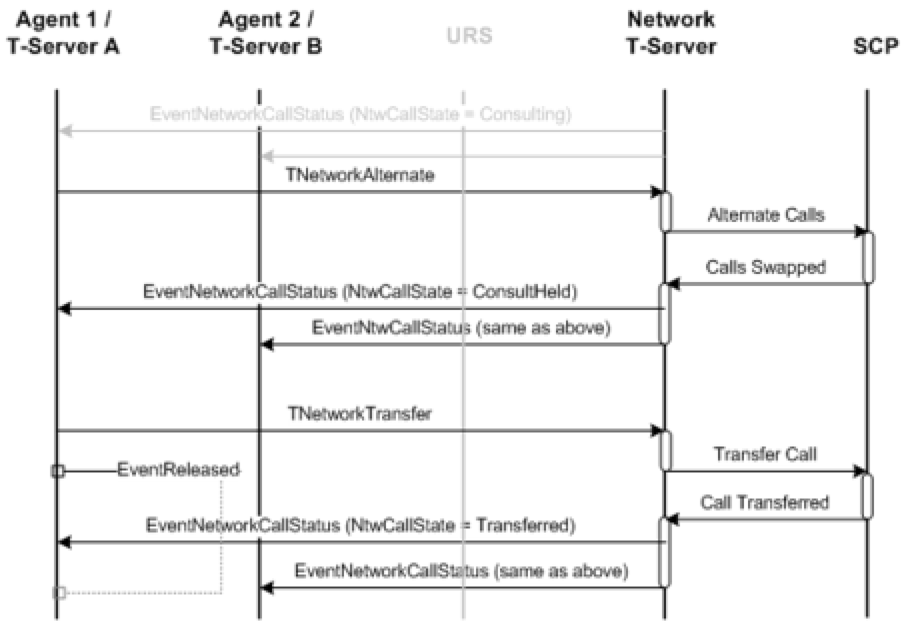
Alternate Call Service

Alternate Call Service with Transfer Completion

The following figure demonstrates the applicability of `TNetworkTransfer` after the use of the alternate call service.

Important

`TNetworkConference`, `TNetworkReconnect`, and the implicit form of transfer completion are also available after an instance of the alternate call service.

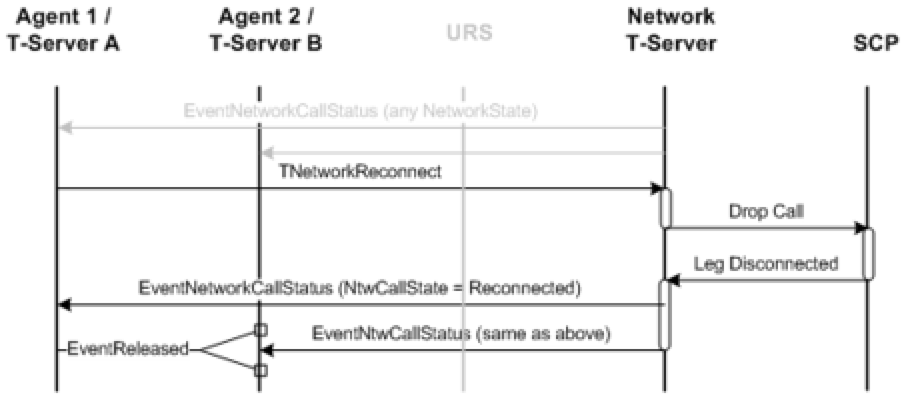

Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
EventNetworkCallStatus (NtwCallState = Consulting)
TNetworkAlternate
Alternate Calls
Calls Swapped
EventNetworkCallStatus (NtwCallState = ConsultHeld)
EventNtwCallStatus (same as above)
TNetworkTransfer
Transfer Call
EventReleased
Call Transferred
EventNetworkCallStatus (NtwCallState = Transferred)
EventNetworkCallStatus (same as above)

Alternate Call Service with Transfer Completion

Reconnection

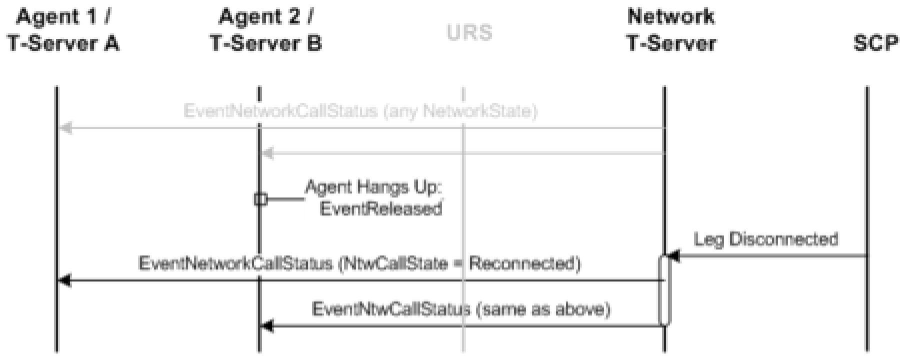
The following three figures show the call flows for reconnecting the caller to Agent 1. As with transfer completion, reconnection has both explicit (Network Attended Reconnection: Explicit) and implied (Network Attended Reconnection: Implicit by SCP and Network Attended Reconnection: Implicit by Network T-Server) forms. In fact, the only difference between the implicit form for network transfer and network reconnection is a party's relationship to the original call. If the controlling agent disconnects, the action is considered a transfer. If the destination agent disconnects, it is considered a reconnection.

Explicit Reconnection


Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
EventNetworkCallStatus (any NetworkState)
TNetworkReconnect
Drop Call
Leg Disconnected
EventNetworkCallStatus (NtwCallState = Reconnected)
EventNtwCallStatus (same as above)
EventReleased

Network Attended Reconnection: Explicit

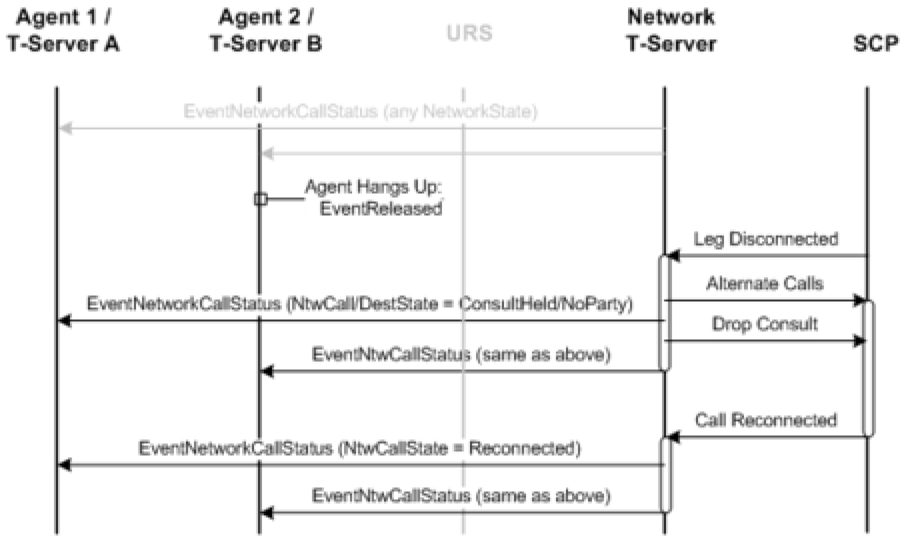
Implicit Reconnection (by SCP)


Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
EventNetworkCallStatus (any NetworkState)
Agent Hangs Up: EventReleased
Leg Disconnected
EventNetworkCallStatus (NtwCallState = Reconnected)
EventNtwCallStatus (same as above)

Network Attended Reconnection: Implicit by SCP

Implicit Reconnection (by Network T-Server)

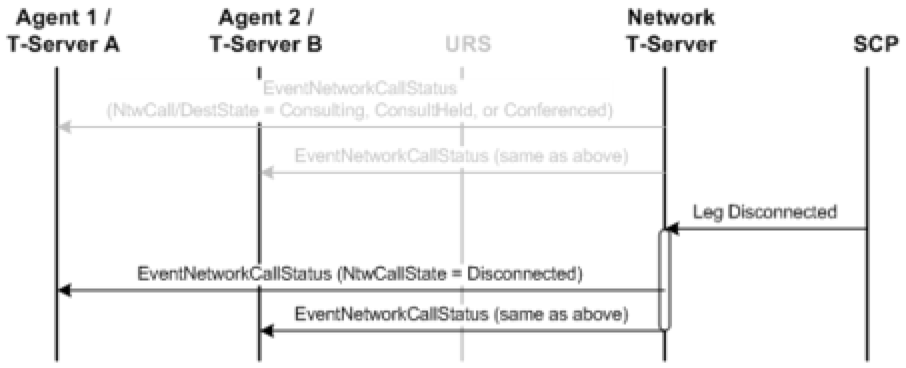
In some cases, if Agent 2 disconnects while the consultation call is on hold, SCP leaves the consultation leg to be dropped manually. Although this operation then becomes the responsibility of the Network T-Server, the client may get a notification regarding an intermediate state, as shown in the following figure.


Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
EventNetworkCallStatus (any NetworkState)
Agent Hangs Up: EventReleased
Leg Disconnected
Alternate Calls
EventNetworkCallStatus (NtwCall/DestState = ConsultHeld/NoParty)
Drop Consult
EventNtwCallStatus (same as above)
Call Reconnected
EventNetworkCallStatus (NtwCallState = Reconnected)
EventNtwCallStatus (same as above)

Network Attended Reconnection: Implicit by Network T-Server

Caller Abandonment

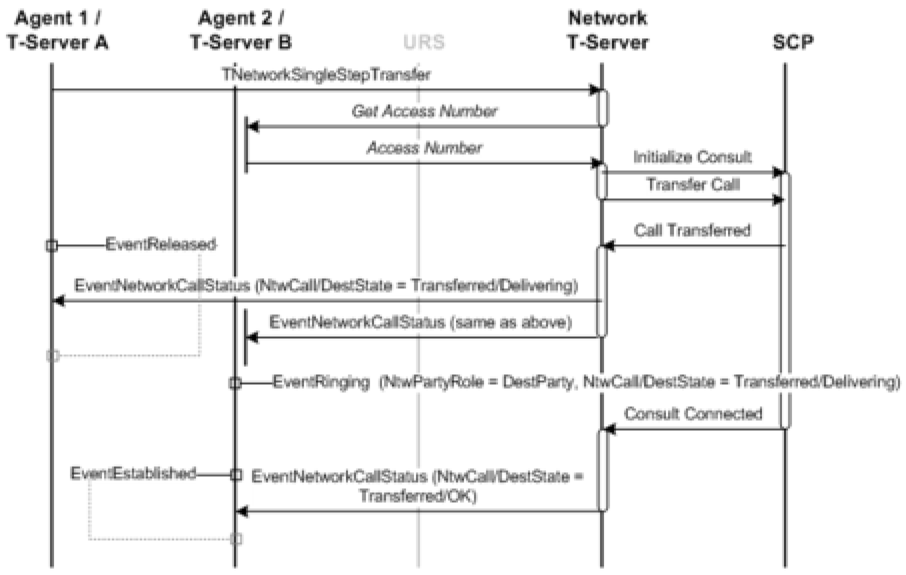
The following figure illustrates a caller abandonment scenario. If at any time during a multi-party call the origination party disconnects, the SCP may choose to end the call outright, or send a leg disconnected message.


Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
EventNetworkCallStatus (NtwCall/DestState = Consulting, ConsultHeld, or Conferenced)
EventNetworkCallStatus (same as above)
Leg Disconnected
EventNetworkCallStatus (NtwCallState = Disconnected)
EventNetworkCallStatus (same as above)

Caller Abandonment

Network Single-Step Transfer

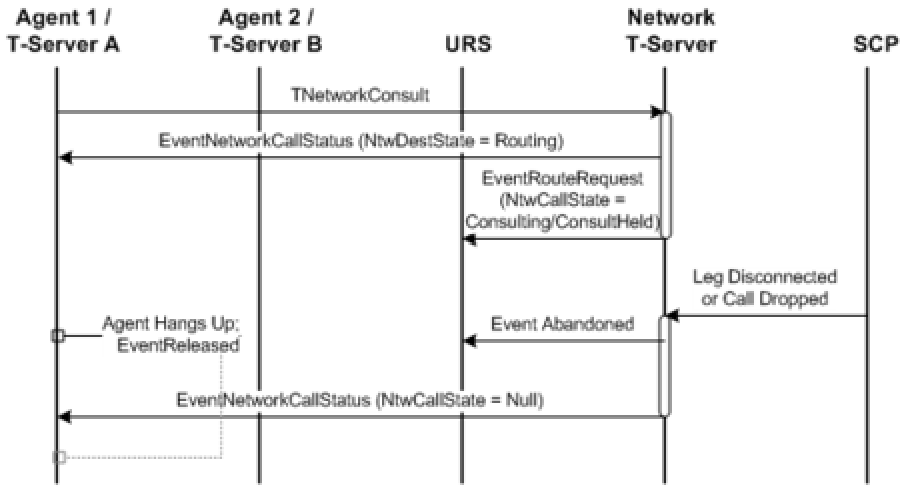
For single-step transfers, the operation is complete immediately after the new call leg is created. Thus, for the external observer, there is no consultation phase.


Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
TNetworkSingleStepTransfer
Get Access Number
Access Number
Initialize Consult
Transfer Call
Call Transferred
EventReleased
EventNetworkCallStatus (NtwCall/DestState = Transferred/Delivering)
EventNetworkCallStatus (same as above)
EventRinging (NtwPartyRole = DestParty, NtwCall/DestState = Transferred/Delivering)
Consult Connected
EventEstablished
EventNetworkCallStatus (NtwCall/DestState = Transferred/OK)

Network Attended Single-Step Transfer

Premature Disconnection, One Variation

If an agent disconnects the call prior to completion of the consultation leg, a number of things can happen, depending on the current state of the consultation. One variation is detailed in the following figure. In this case, the disconnection occurs prior to the SCP being aware of a pending consultation. The message from the SCP is likely to be `Call Dropped`, as only the caller would remain on the call.

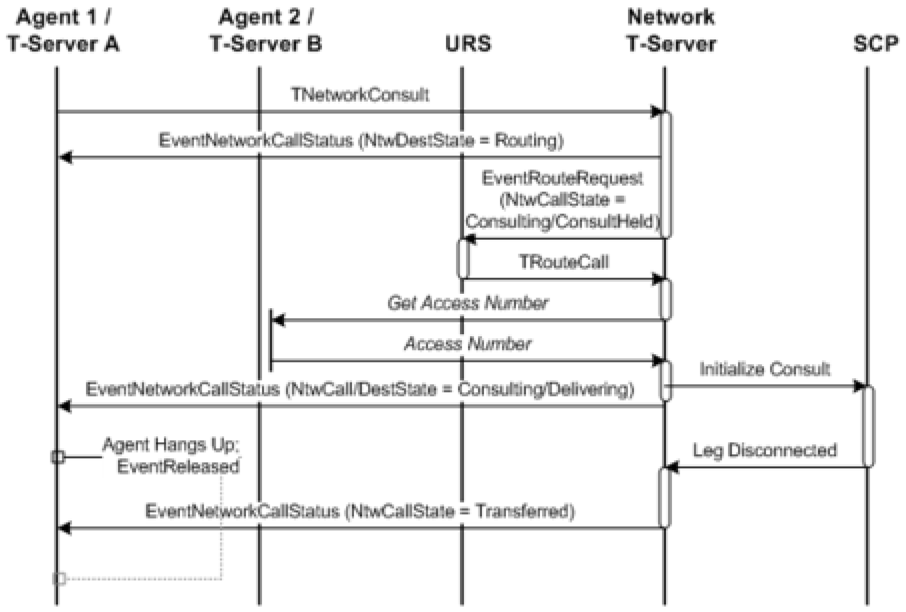

Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
TNetworkConsult
EventNetworkCallStatus (NtwDestState = Routing)
EventRouteRequest (NtwCallState = Consulting/ConsultHeld)
Leg Disconnected or Call Dropped
Agent Hangs Up; EventReleased
Event Abandoned
EventNetworkCallStatus (NtwCallState = Null)

Premature Disconnection: One Variation

Premature Disconnection, a Second Variation

The following figure shows Agent 1 disconnecting after the SCP has been notified of a call, but prior to its connection (or failure). Since the state of the consultation is not known, the call considered to have transferred. The SCP at this point should connect the original caller with the consultation target. The call at this point is treated identically to a normal inbound call that is waiting for its connection status from the SCP. The network attended session is complete, and no further `NetworkCallStatus` events are distributed.

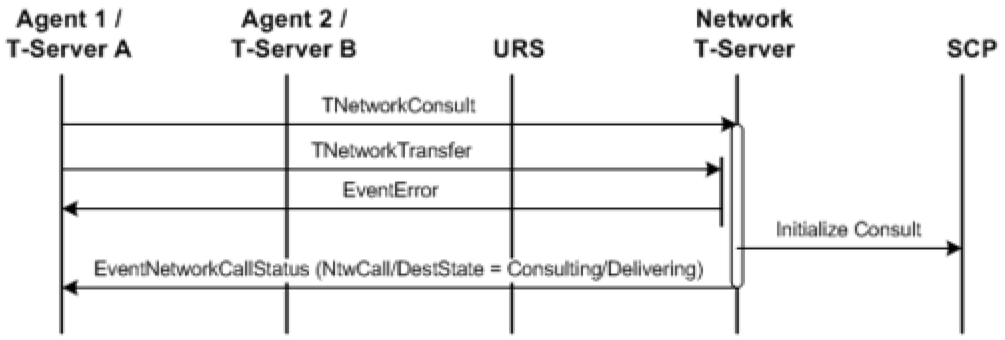
If the delivery results in failure (or a `NoAnswer` condition), URS should re-route the call to a different target. However, if the call was intended for an explicitly specified target (and URS is not in control of the consultation leg), any call status other than `Connected` results in default routing instructions being returned to the SCP, and the call ending.


Agent 1 / T-Server A
Agent 2 / T-Server B
URS
Network T-Server
SCP
TNetworkConsult
EventNetworkCallStatus (NtwDestState = Routing)
EventRouteRequest (NtwCallState = Consulting/ConsultHeld)
TRouteCall
Get Access Number
Access Number
Initialize Consult
EventNetworkCallStatus (NtwCall/DestState = Consulting/Delivering)
Agent Hangs Up; EventReleased
Leg Disconnected
EventNetworkCallStatus (NtwCallState = Transferred)

Premature Disconnection: a Second Variation

Transactional Error

A T-Library client is required to wait for a network status message prior to attempting further call control. This message is either the next status message after making its network request or `EventError`. Failure to abide by this results in an error being returned to the requestor.



Transactional Error

Shared Call Appearance

Introduced in SIP Server version 8.1.101.57, Shared Call Appearance (SCA) enables a group of SIP phones to receive inbound calls directed to a single destination (shared line); that way, any phone from this group can answer the call, barge-in to the active call, or retrieve the call placed on hold.

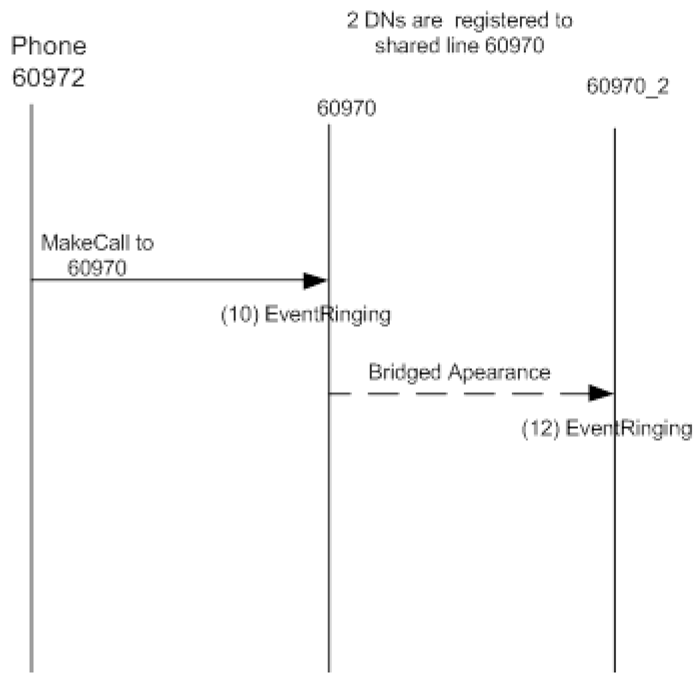
The call flows in this section describe the functions and events related to SCA functionality:

- Inbound Call Alerting
- Inbound Call Answered
- Call Termination by a Non-shared Line Phone
- Outbound Call from SCA
- Hold/Retrieve
- Barge-in

Note: Subscription to other T-Library events is required to see actions on the calls and for Barge-in/Retrieve functions. The T-Library client must not allow to send 3pcc requests from a shared-line user for the calls that are owned by other T-Library clients.

Inbound Call Alerting

The following graphic and table describe the inbound call alerting.

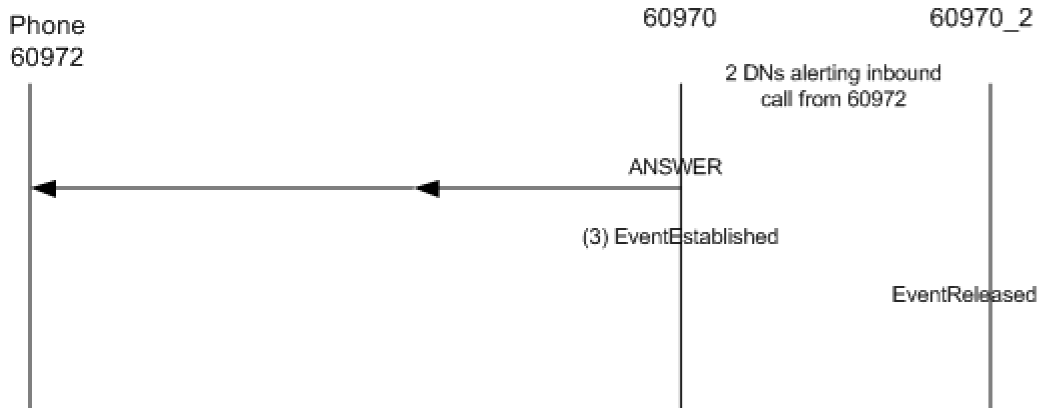


Inbound Call Alerting

Caller (60972)	DN 60970	DN 60970_2	Call Monitoring
EventOffhook ThisDN **60972** **EventDialing** ConnID **1** ThisDN **60972** OtherDN **60970** CallState **Ok**	**(10) EventRinging** ConnID **1** ThisDN **60970** ThisDNRole **Destination** OtherDN **60972** OtherDNRole **Origination** CallState **Covered** CallType **Internal/Inbound** (depends on 60972 location) Extensions: AppearanceIndex **1** CallState value is **Covered** because this is a call to a shared line.	**(12) EventRinging** ConnID **1** ThisDN **60970_2** ThisDNRole **Destination** OtherDN **60972** OtherDNRole **Origination** CallState **Covered** CallType **Internal/Inbound** (depends on 60972 location) Extensions: AppearanceIndex **1** CallState value is **Covered** because this is a call to a shared line.	**EventCallCreated** ConnID **1** **EventCallPartyAdded** ConnID **1** DN **60972** PartyID **1** State **Initiated** **EventCallPartyState** ConnID **1** PartyID **1** State **Connected+Dialing** **EventCallPartyAdded** ConnID **1** DN **60970** PartyID **2** State **Alerting** **EventCallPartyAdded** ConnID **1** DN **60970_2** PartyID **3** State **Alerting**

Inbound Call Answered

The following graphic and table describe the inbound call answered call flow.

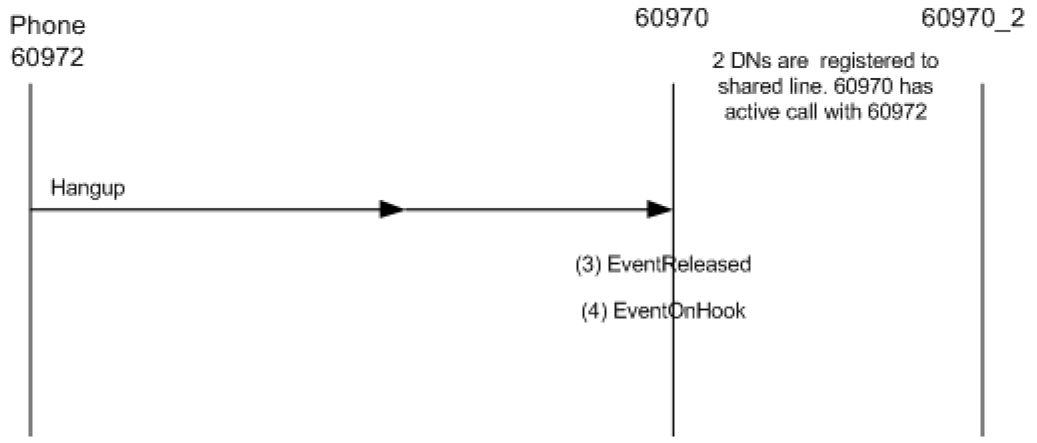


Inbound Call Answered

Caller (60972)	DN 60970	DN 60970_2	Call Monitoring
EventEstablished ConnID **1** ThisDN **60972** ThisDNRole **Origination** OtherDN **60970** OtherDNRole **Destination**	**(3) EventEstablished** ConnID **1** ThisDN **60970** ThisDNRole **Destination** OtherDN **60972** OtherDNRole **Origination** Extensions: AppearanceIndex **1**	**EventReleased** ConnID **1** ThisDN **60970_2** ThisDNRole **Destination** OtherDN **60972** OtherDNRole **Origination** CallState **Redirected**	**EventCallPartyDeleted** ConnID **1** PartyID **3** (60970_2) **EventCallPartyState** ConnID **1** PartyID **2** State **Connected**

Call Termination by a Non-shared Line Phone

The following graphic and table describe the call termination by a non-shared line phone call flow.


Phone 60972
60970
60970_2
2 DNs are registered to shared line. 60970 has active call with 60972
Hangup
(3) EventReleased
(4) EventOnHook

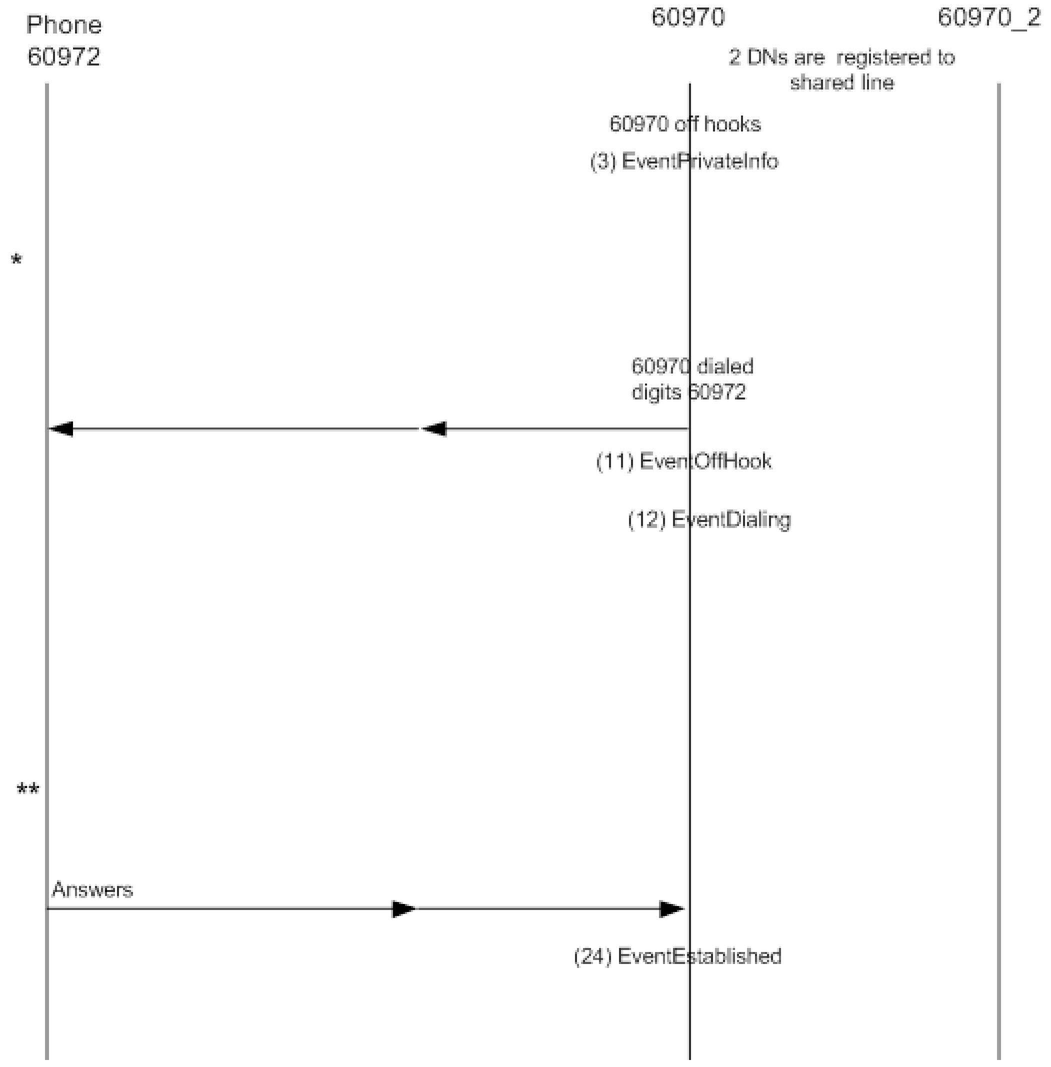
Call Termination by a Non-shared Line Phone

Caller (60972)	DN 60970	DN 60970_2	Call Monitoring
EventReleased	**(3) EventReleased**		**EventCallPartyDeleted**

ConnID **1** ThisDN **60972** OtherDN **60970**	ConnID **1** ThisDN **60970** OtherDN **60972** CallState **OK** **(4) EventOnHook** ThisDN **60970**		ConnID **1** PartyID **2** **EventCallPartyDeleted** ConnID **1** PartyID **1** **EventCallDeleted** ConnID **1**

Outbound Call from SCA

The following graphic and table describe the outbound call from SCA call flow.



Outbound Call from SCA

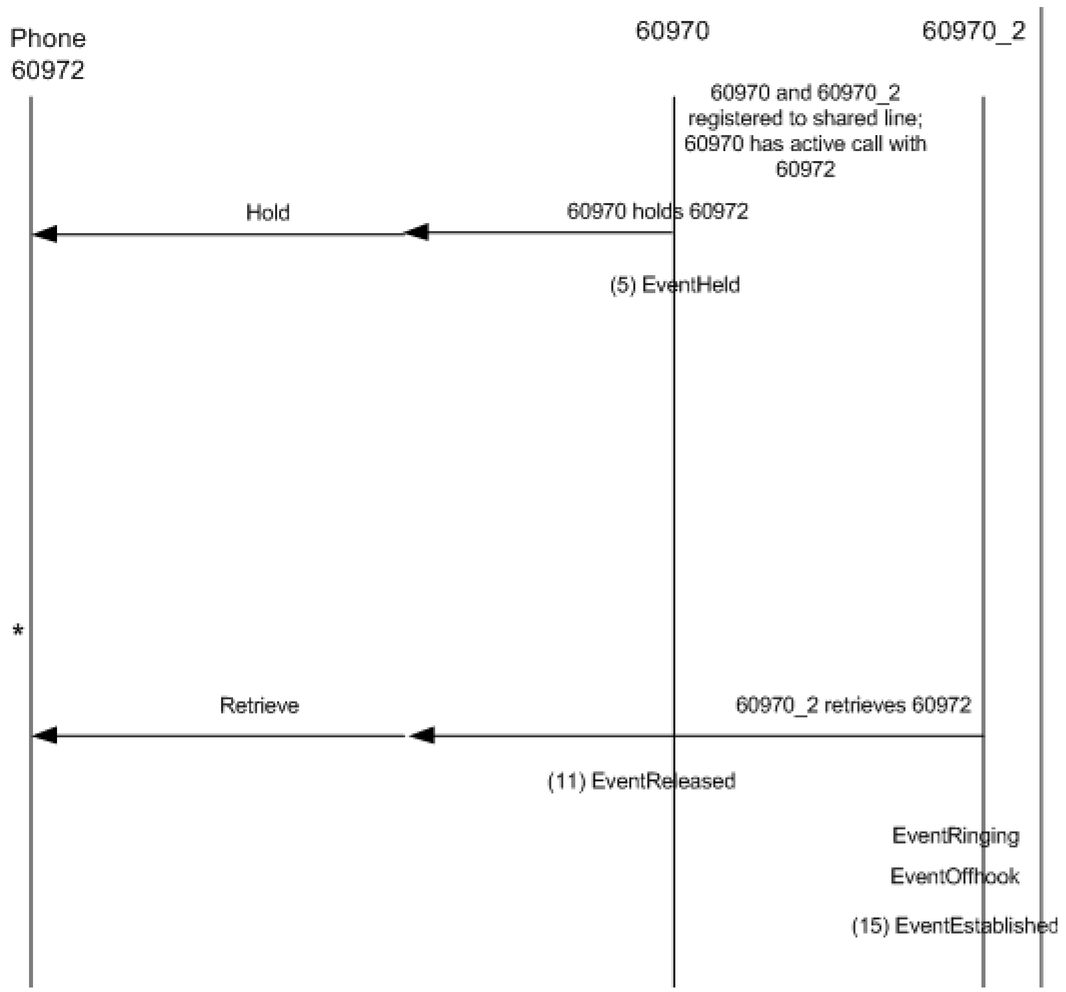
Caller (60972)	DN 60970	DN 60970_2	Call Monitoring
EventRinging OtherDN **60970** **EventEstablished** OtherDN **60970**	**(3) EventPrivateInfo** ServiceId **4026** ThisDN **60970** Extensions: AppearanceIndex **1** **(11) EventOffHook** ThisDN **60970** Extensions: AppearanceIndex **1** **(12) EventDialing** ConnID **1** ThisDN **60970** ThisDNRole **Origination** OtherDN **60972** OtherDNRole **Destination** Extensions: AppearanceIndex **1** **(24) EventEstablished** ConnID **1** ThisDN **60970** ThisDNRole **Origination** OtherDN **60972** OtherDNRole **Destination** Extensions: AppearanceIndex **1**		**EventCallCreated** ConnID **1** **EventCallPartyAdded** ConnID **1** DN **60970** PartyID **1** State **Initiated** **EventCallPartyState** ConnID **1** PartyID **1** State **Connected+Dialing** **EventCallPartyAdded** ConnID **1** DN **60972** PartyID **3** State **Alerting** **EventCallPartyState** ConnID **1** PartyID **3** (60972) State **Connected**

Abnormal Call Flow

Interruption Point	DN 60970	DN 60970_2	Call Monitoring
* 60970 hangs up	**EventReleased** ConnID **1** ThisDN **60970** OtherDN **60972** CallState **OK** **EventOnHook** ThisDN **60970**		
** 60970 hangs up	**EventReleased** ConnID **1** ThisDN **60970** OtherDN **60972** CallState **OK**		**EventCallPartyDeleted** ConnID **1** PartyID **1** **EventCallDeleted** ConnID **1**

Hold/Retrieve

The following graphic and table describe the hold/retrieve call flow.



Hold/Retrieve

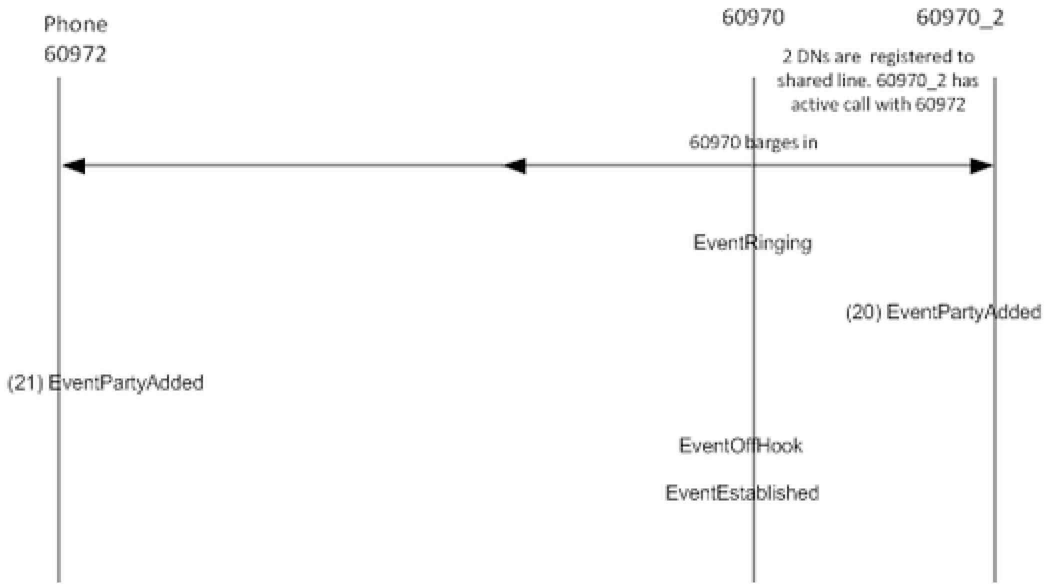
DN 60970	DN 60970_2	Call Monitoring
(5) EventHeld ConnID **1** ThisDN **60970** OtherDN **60972** **(11) EventReleased** ConnID **1** ThisDN **60970** OtherDN **60972** CallState **Transferred** Cause **1stepTransfer**	**EventOffhook** ThisDN **60970_2** Extensions: AppearanceIndex **1** **EventRinging** ConnID **1** ThisDN **60970_2** OtherDN **60972** CallState **Transferred** Extensions: AppearanceIndex **1** **(15) EventEstablished** ConnID **1** ThisDN **60970_2** OtherDN **60972** CallState **OK** Extensions: AppearanceIndex **1**	**EventCallPartyState** ConnID **1** PartyID **60970partyID** State **Held** **EventCallPartyDeleted** ConnID **1** PartyID **60970PartyID** **EventCallPartyAdded** ConnID **1** PartyID **60970_2partyID** State **Alerting** **EventCallPartyState** ConnID **1** PartyID **60970_2partyID** State **Connected**

Abnormal Call Flow

Interruption Point	DN 60970	DN 60970_2	Call Monitoring
* 60970 hangs up	**EventReleased** ConnID **1** ThisDN **60970** OtherDN **60972** CallState **OK**		**EventCallPartyDeleted** ConnID **1** PartyID **60970PartyID** **EventCallPartyDeleted** ConnID **1** PartyID **60972PartyID** **EventCallDeleted** ConnID **1**

Barge-in

The following graphic and table describe the barge-in call flow.



Barge-in

	DN 60970	DN 60970_2	Call Monitoring
EventPartyAdded ConnID **1** ThisDN **60972** OtherDN **60970** OtherDNRole **NewParty** ThirdPartyDN **60970** ThirdPartyDNRole **AddedBy** CallState **Conferenced**	**EventRinging** ConnID **1** ThisDN **60970** CallState **OK** ThisDNRole **ConferenceMember** Extensions: AppearanceIndex **1**	**EventPartyAdded** ConnID **1** ThisDN **60970_2** OtherDN **60970** OtherDNRole **NewParty** ThirdPartyDN **60970** ThirdPartyDNRole **AddedBy** CallState **Conferenced**	**EventCallPartyAdded** ConnID **1** PartyID **60970partyID** State **Alerting**

	EventOffhook ThisDN **60970** Extensions: AppearanceIndex **1** **EventEstablished** ConnID **1** ThisDN **60970** CallState **Conferenced** Extensions: AppearanceIndex **1**		**EventCallPartyState** ConnID **1** PartyID **60970partyID** State **Connected**

Basic Interaction Models

This section presents models representing basic tasks that the Multimedia components perform in terms of scenarios. Some scenarios are made up of a single model. Others consist of several models that represent different temporal phases. Others collect several models that represent different but related versions of a single general scenario.

The events that occur in each model are all documented in Open Media Interaction Models documentation.

This section has the following subsections:

Registration

This set of models illustrates successful and unsuccessful registration. Successful registration proceeds as follows:

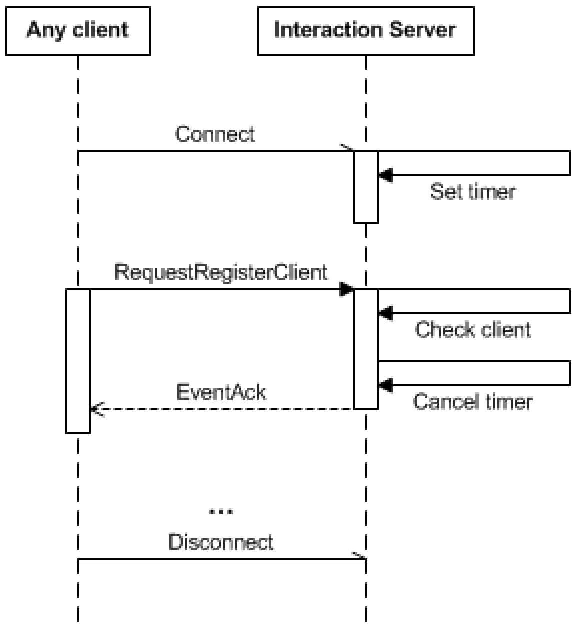
1. A client connects to Interaction Server.
2. The client asks to register.
3. Interaction Server checks to see if the client is valid. If the client is valid, Interaction Server sends `EventAck`.

Unsuccessful registration proceeds as follows:

1. A client connects to Interaction Server.
2. One of the following happens:
   - The client asks to register, but Interaction Server finds that it is not a valid client.
   - The client sends any other message to Interaction Server.
3. In either case, Interaction Server returns `EventError`.
4. When a timeout expires, Interaction Server disconnects.

Successful Registration

In this phase, shown in the following figure, a client connects, then asks to register.



Successful Registration

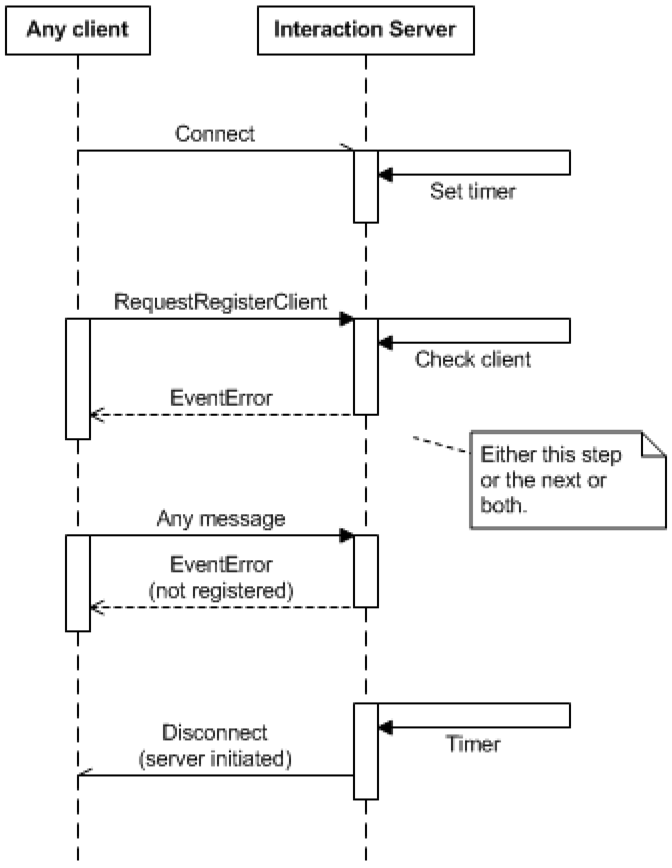
This phase uses the messages listed in the following table:

Messages in Successful Registration

Message	Protocol
EventAck	Interaction Management

Unsuccessful Registration

In this phase, shown in the following figure, Interaction Server finds that the client is not valid. It may do this in response to `RequestRegisterClient` or to any other message from the client. In any case, Interaction Server responds with `EventError`, then disconnects from the client.



Unsuccessful Registration

The figure above actually contains three possible versions:

1. Client sends `RequestRegisterClient`, Interaction Server finds that the client is not valid, Interaction server returns `EventError`.
2. Client sends any message Interaction Server finds that the client is not registered, Interaction server

returns `EventError`.

3. First a, then b.

This phase uses the messages listed in the following table:

Messages in Unsuccessful Registration

Message	Protocol
EventError	Interaction Management

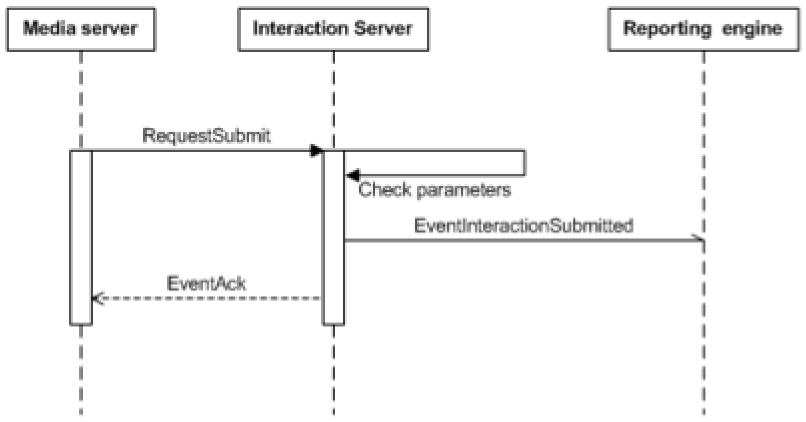
Media Server Submits Interaction

This set of models illustrates the following scenario:

1. A media server asks to submit an interaction to Interaction Server. Interaction Server checks the interaction's parameters. If the parameters are correct, Interaction Server accepts the request.
2. If Interaction Server discovers that the parameters are incorrect, it rejects the request.

Media Server Asks to Submit

In this phase, shown in the following figure, a media server sends `RequestSubmit` to Interaction Server.



Media Server Asks to Submit

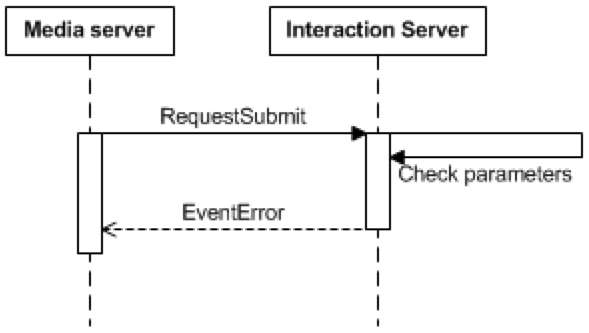
This phase uses the messages listed in the following table:

Messages in Media Server Asks to Submit

Message	Protocol
EventAck	Interaction Management
EventInteractionSubmitted	Reporting

Unsuccessful Submission by Media Server

In this phase, shown in the following figure, Interaction Server discovers a problem, such as that the media server is not registered as a client, or that there is a error in one of the attributes of `RequestSubmit.`



Unsuccessful Submission by Media Server

This phase uses the messages listed in the following table:

Messages in Unsuccessful Submission by Media Server

Message	Protocol
EventError	Interaction Management

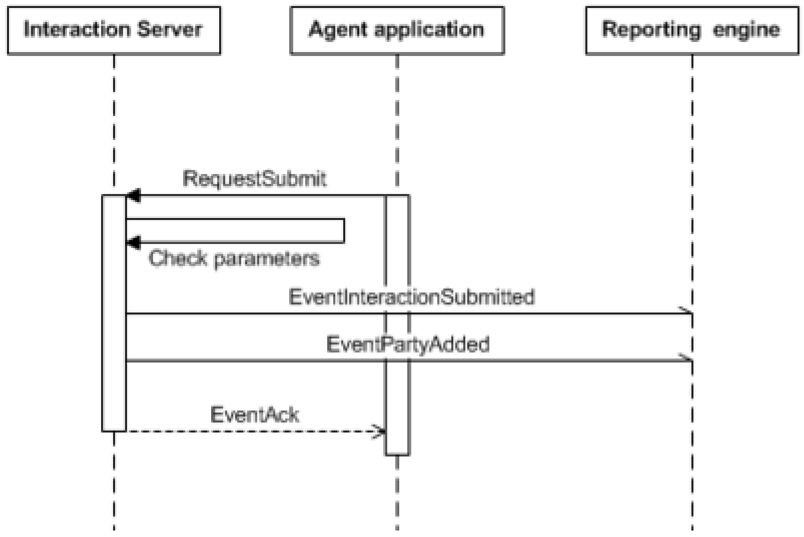
Agent Submits Interaction

This set of models illustrates the following scenario:

1. An agent asks to submit an interaction.
2. Interaction Server checks the validity of the agent application.
3. One of the following happens:
    - If the agent application is valid, Interaction Server accepts the submission and informs the reporting engine.
    - If the agent application is not valid, Interaction Server rejects the submission.

Successful Submission

A successful submission is shown in the following figure:



Successful Submission by Agent

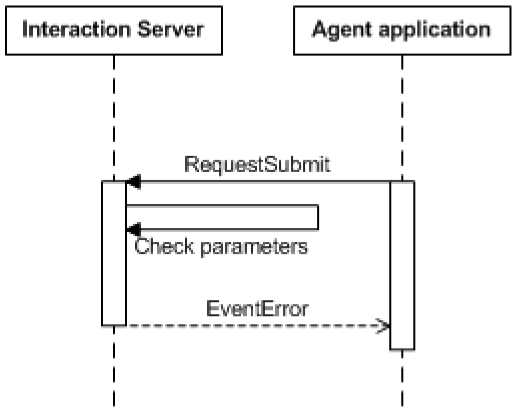
This model uses the messages listed in the following table:

Messages in Successful Submission by Agent

Message	Protocol
EventAck	Interaction Management
EventInteractionSubmitted	Reporting
EventPartyAdded	Reporting

Unsuccessful Submission

An unsuccessful submission is shown in the following figure:



Unsuccessful Submission by Agent

This model uses the messages listed in the following table:

Messages in Unsuccessful Submission by Agent

Message	Protocol
EventError	Interaction Management

Stop Processing

This scenario has several versions, depending on these two points:

- Which entity initiates the stop processing request: media server, agent, or URS
- If a media server initiates the request, there is a further difference according to whether an agent or URS is involved in processing.

This produces four possible versions, called A, B, C, and D in this section:

A—Initiated by media server, agent involved in processing

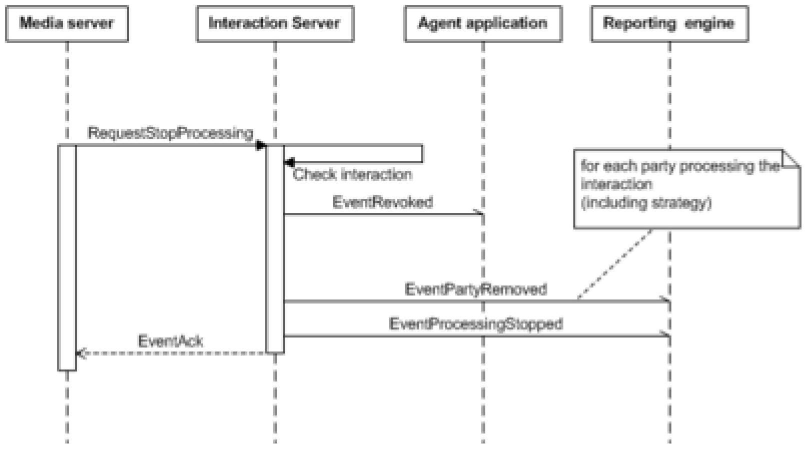
B—Initiated by media server, URS involved in processing

C—Initiated by agent

D—Initiated by URS

A: Initiated by Media Server, Agent Involved

In this version, shown in the following figure, an agent is processing the interaction when a media server asks for processing to stop.



Stop Processing A: Media Server Initiates, Agent Involved

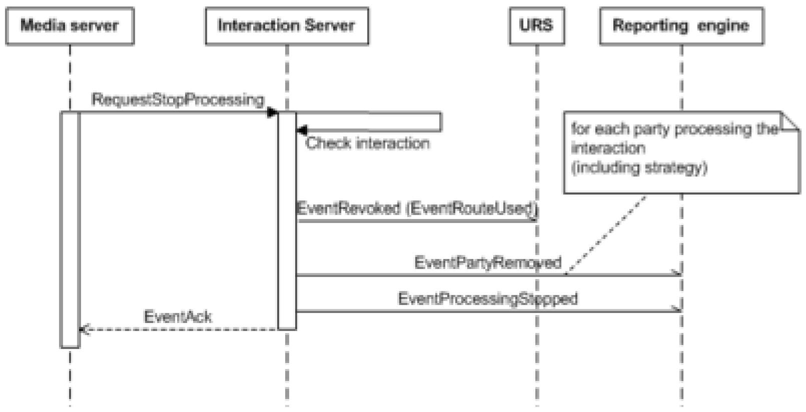
This version uses the messages listed in the following table:

Messages in Stop Processing A

Message	Protocol
EventAck	Interaction Management
EventPartyRemoved	Reporting
EventProcessingStopped	Reporting
EventRevoked	Interaction Management

B: Initiated by Media Server, URS Involved

In this version, shown in the following figure, URS is processing the interaction when a media server asks for processing to stop.



Stop Processing B: Media Server Initiates, URS Involved

This version uses the messages listed in the following table:

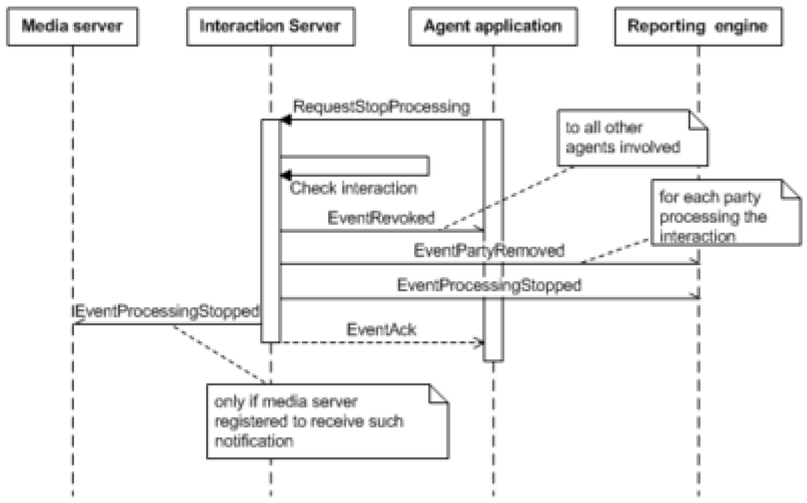
Messages in Stop Processing B

Message	Protocol
EventAck	Interaction Management
EventPartyRemoved	Reporting
EventProcessingStopped	Reporting
EventAbandoned, used here in place of EventRevoked	T-Library

This model uses the T-Library `EventRouteUsed` to stand in for the Interaction Management `EventRevoked`.

C: Initiated By Agent

In this version, shown in the following figure, an agent initiates the stop processing request.



Stop Processing C: Initiated by Agent

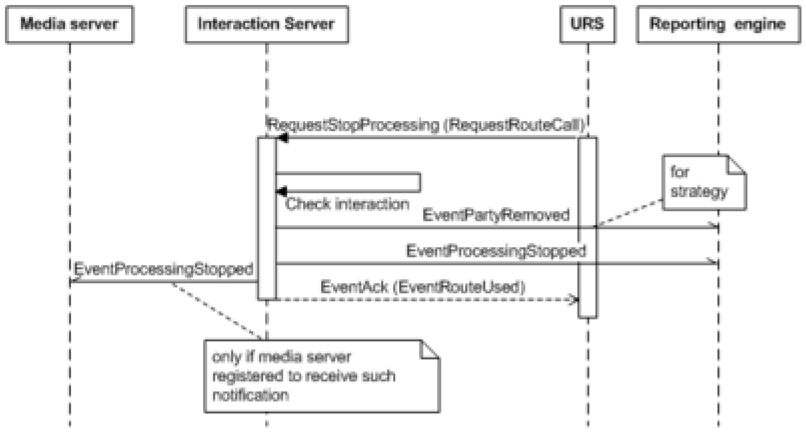
This version uses the messages listed in the following table:

Messages in Stop Processing C

Message	Protocol
EventAck	Interaction Management
EventPartyRemoved	Reporting
EventProcessingStopped	Reporting
EventRevoked	Interaction Management

D: Initiated by URS

In this version, shown in the following figure, URS initiates the stop processing request.



Stop Processing D: Initiated by URS

This version uses the messages listed in the following table:

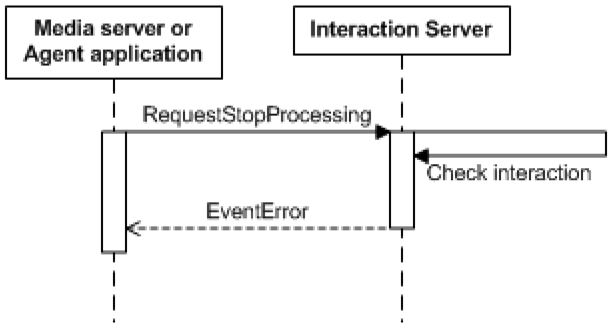
Messages in Stop Processing D

Message	Protocol
EventAck	Interaction Management
EventPartyRemoved	Reporting
EventProcessingStopped	Reporting
EventRouteUsed, used here in place of EventAck	T-Library

This phase uses the T-Library `RequestRouteCall` to stand in for the Interaction Management `RequestStopProcessing`. It also uses the T-Library `EventRouteUsed` to stand in for the Interaction Management `EventAck`.

Unsuccessful

In this version, shown in the following figure, Interaction Server finds that the request is invalid and rejects it.



Stop Processing: Unsuccessful

This version uses the messages listed in the following table:

Messages in Stop Processing: Unsuccessful

Message	Protocol
EventError	Interaction Management

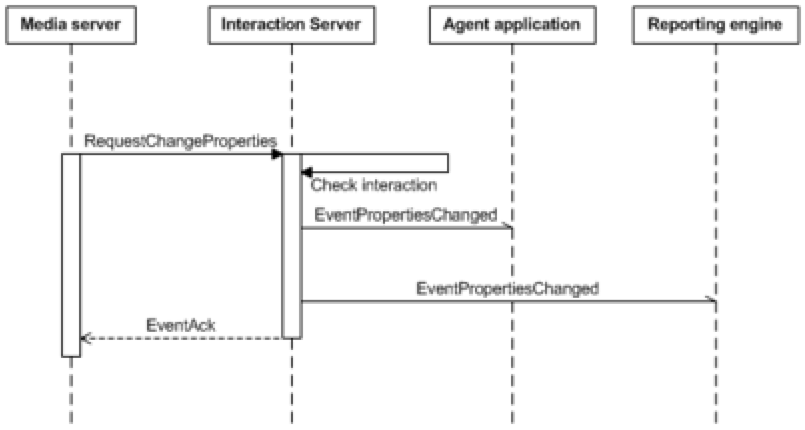
Change Properties

This scenario has several versions, differentiated first of all according to which entity initiates the request to change properties:

- Media server initiates the request
    - While an agent is processing the interaction.
    - While URS is processing the interaction.
- URS initiates the request (only while URS is processing).

Media Server Requests While Agent is Processing

In this phase, shown in the following figure, a media server asks to change the properties of an interaction. Interaction Server informs the reporting engine and the agent who is processing the interaction.



Media Server Requests While Agent is Processing

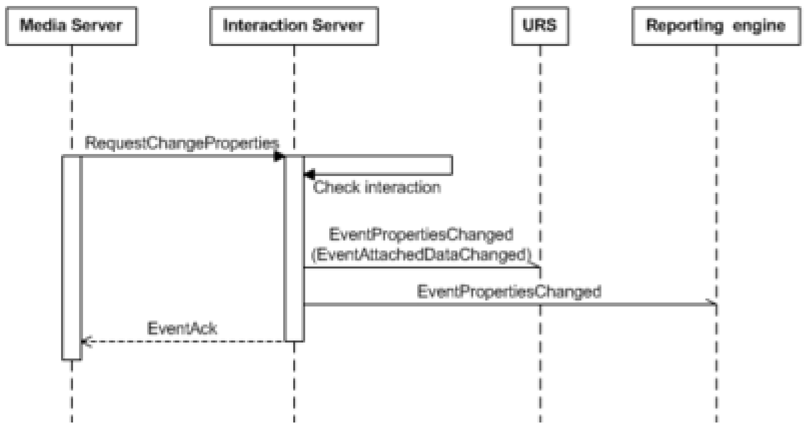
This phase uses the messages shown in the following table:

Messages in Media Server Requests While Agent is Processing

Message	Protocol
EventAck	Interaction Management
EventPropertiesChanged	Interaction Management

Media Server Requests While URS is Processing

In this phase, shown in the following figure, a media server asks to change the properties of an interaction. Interaction Server informs the reporting engine and URS, which is processing the interaction.



Media Server Requests While URS is Processing

This phase uses the messages shown in the following table:

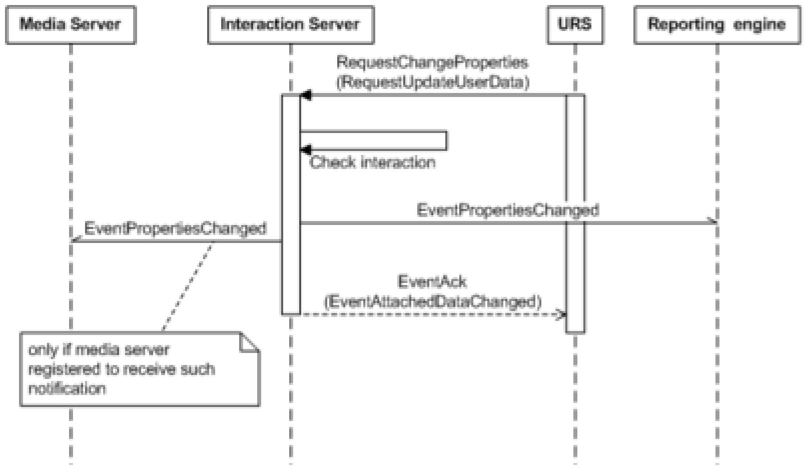
Messages in Media Server Requests, URS is Processing

Message	Protocol
EventAck	Interaction Management
EventAttachedDataChanged, used here in place of EventPropertiesChanged	T-Library

This phase uses the T-Library `EventAttachedDataChanged` to stand in for the Interaction Management `EventPropertiesChanged`.

URS Requests

In this phase, shown in the following figure, URS changes the interaction properties. If the media server has included the extension `event_properties_changed`, with nonzero value, in its `RequestRegisterClient` message, then Interaction Server sends `EventProperitesChanged` informing it of the change.


Media Server
Interaction Server
URS
Reporting engine
RequestChangeProperties (RequestUpdateUserData)
Check interaction
EventPropertiesChanged
EventPropertiesChanged
EventAck (EventAttachedDataChanged)
only if media server registered to receive such notification

URS Requests

This phase uses the messages shown in the following table:

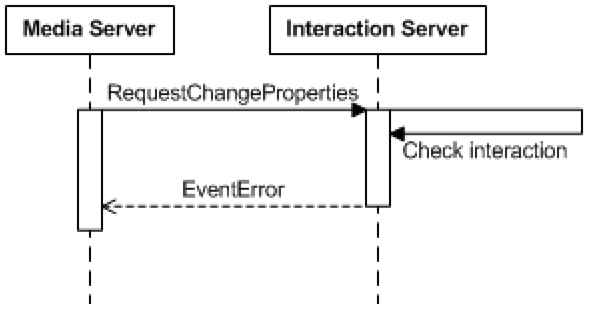
Messages in URS Requests

Message	Protocol
EventAttachedDataChanged, used here in place of EventAck	T-Library
EventPropertiesChanged	Interaction Management

This phase uses the T-Library `RequestUpdateUserData` to stand in for the Interaction Management `RequestChangeproperties`. It also uses the T-Library `EventAttachedDataChanged` to stand in for the Interaction Management `EventAck`.

Unsuccessful Request

In this phase, shown in the following figure, Interaction Server finds that the request is invalid.



Unsuccessful Request

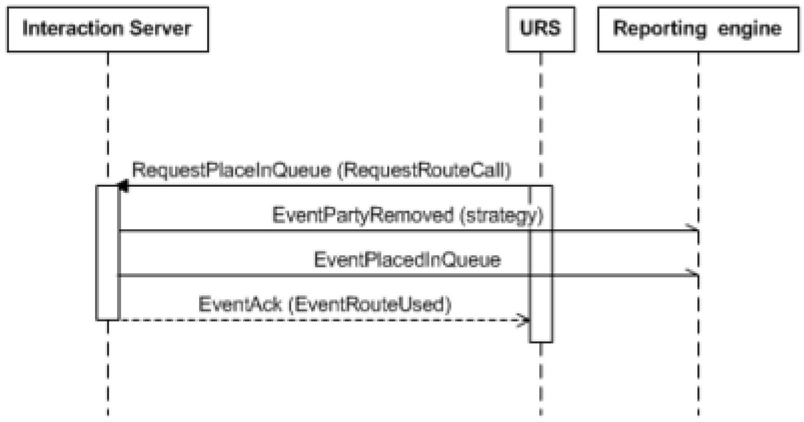
This phase uses the messages shown in the following table:

Messages in Unsuccessful

Message	Protocol
EventError	Interaction Management

Place Interaction in Queue

In this model, shown in the following figure, URS requests Interaction Server to place an interaction in a queue.


Interaction Server
URS
Reporting engine
RequestPlaceInQueue (RequestRouteCall)
EventPartyRemoved (strategy)
EventPlacedInQueue
EventAck (EventRouteUsed)

Place Interaction in Queue

Important

The agent application can also make the request to place an interaction in a queue. In that case the model would look the same, except that Interaction Server would simply send `EventAck` rather than `EventRouteUsed`.

This model uses the messages listed in the following table:

Messages in Place Interaction in Queue

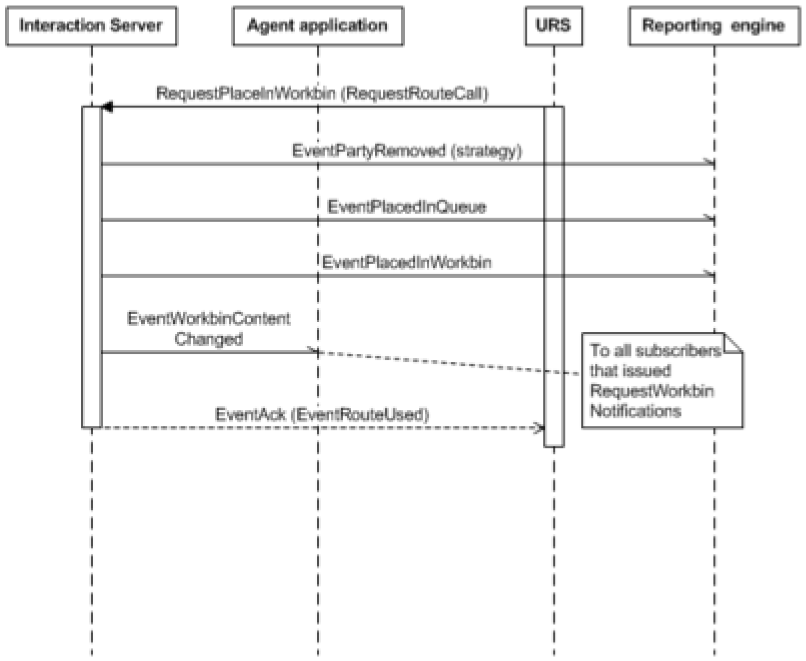
Message	Protocol
EventPartyRemoved	Reporting
EventPlacedInQueue	Reporting
EventRouteUsed, used here in place of EventAck	T-Library

This model uses the following substitutions:

- T-Library `EventRouteUsed` stands in for Interaction Management `EventAck`
- T-Library `RequestRouteCall` stands in for Interaction Management `RequestPlaceInQueue`.

Place Interaction in Workbin

In this model, shown in the following figure, URS asks Interaction Server to place an interaction in a workbin.


Interaction Server
Agent application
URS
Reporting engine
RequestPlaceInWorkbin (RequestRouteCall)
EventPartyRemoved (strategy)
EventPlacedInQueue
EventPlacedInWorkbin
EventWorkbinContent Changed
To all subscribers that issued RequestWorkbin Notifications
EventAck (EventRouteUsed)

Place Interaction in Workbin

Important

The agent application can also make the request to place an interaction in a workbin. In that case the model would look the same, except that Interaction Server would simply send `EventAck` rather than `EventRouteUsed`.

This model uses the messages listed in the following table:

Messages in Place Interaction in Workbin

Message	Protocol
EventPartyRemoved	Reporting
EventPlacedInQueue	Reporting
EventPlacedInWorkbin	Reporting
EventRouteUsed, used here in place of EventAck	T-Library
EventWorkbinContentChanged	Interaction Management

This model uses the following substitutions:

- T-Library `EventRouteUsed` stands in for Interaction Management `EventAck`.
- T-Library `RequestRouteCall` stands in for Interaction Management `RequestPlaceInWorkbin`.

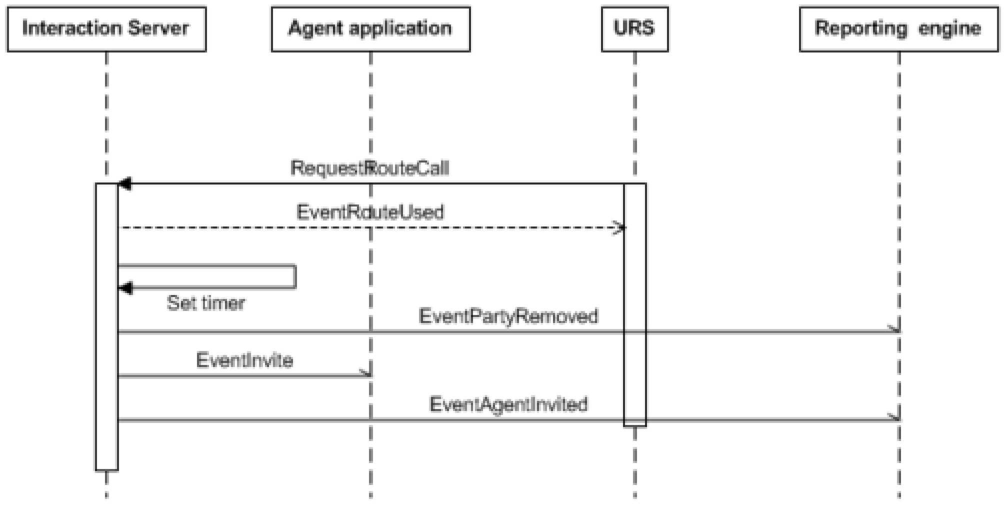
Deliver Interaction to Agent

This set of models illustrates the following scenario:

1. URS asks Interaction Server to attempt to deliver an interaction to an agent.
2. Then one of the following happens:
    - The agent accepts the interaction.
    - The agent rejects the interaction.
    - The agent fails to respond.

URS Requests Delivery

In this phase, shown in the following figure, URS sends `RequestRouteCall` to Interaction Server, specifying the agent and place to receive the interaction. Then Interaction Server sends `EventInvite` to the agent application and sets a timer.



URS Requests Delivery

This phase uses the messages listed in the following table:

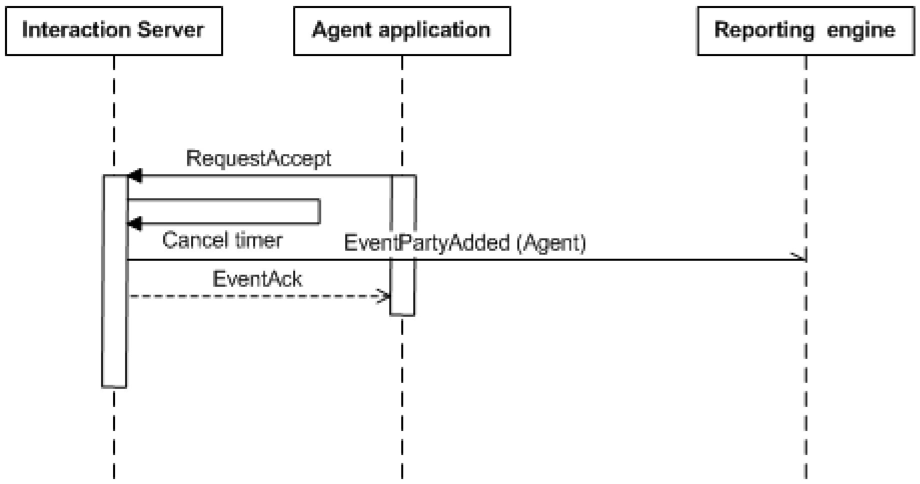
Messages in URS Requests Delivery

Message	Protocol
EventAgentInvited	Reporting
EventInvite	Interaction Management
EventPartyRemoved	Reporting
EventRouteUsed	T-Library

The second phase of this scenario can have one of the following three forms.

Agent Accepts Delivery

In this version of the second phase, shown in the following figure, the agent application accepts delivery of the interaction, and Interaction Server sends EventRouteUsed to URS, informing it that its Deliver request has been filled. Interaction Server also cancels the timer that it started in the previous phase.



Agent Accepts Delivery

This phase uses the messages listed in the following table:

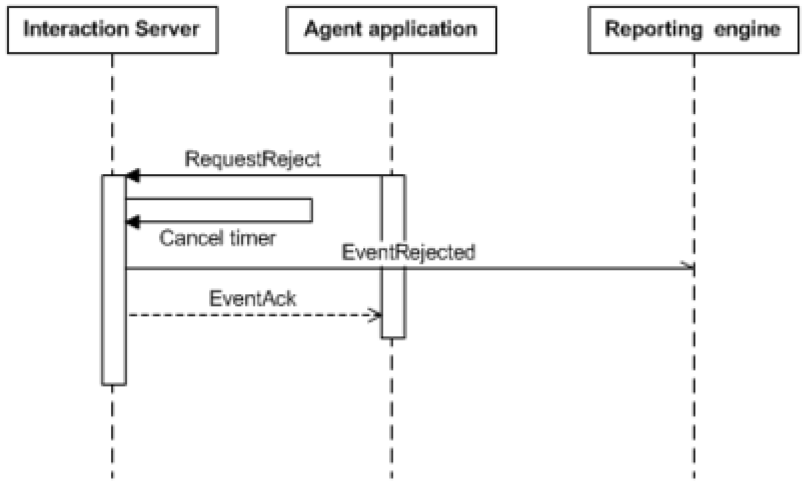
Messages in Agent Accepts Delivery

Message	Protocol
EventPartyAdded	Reporting

There are two ways that the delivery attempt can fail, shown in the next sections.

Agent Rejects Delivery

In this version of the second phase, rather than accepting the interaction as in the figure above, the agent rejects the interaction using `RequestReject`, as shown in the following figure. Interaction Server cancels the timer, acknowledges `RequestReject` using `EventAck`, and informs URS of the situation with `EventRouteUsed`.



Agent Rejects Delivery

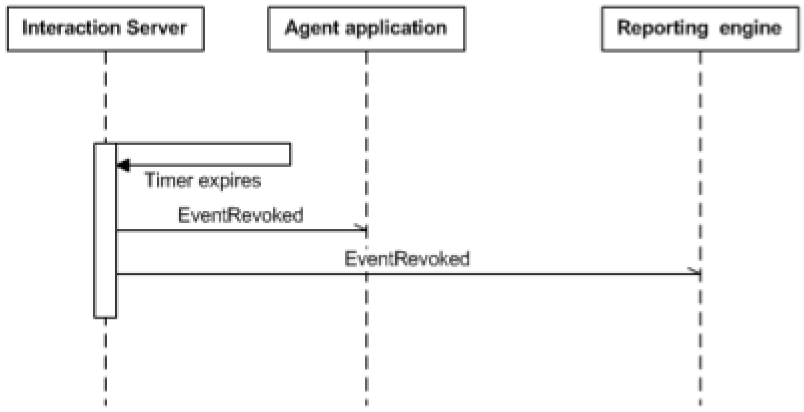
This phase uses the messages listed in the following table:

Messages in Agent Rejects Delivery

Message	Protocol
EventAck	Interaction Management
EventRejected	Reporting

Agent Fails to Respond in Time

In the first phase of this scenario, Interaction Server set a timer. In this version of the second phase, shown in the following figure, the agent application does not respond within the time set and Interaction Server revokes the interaction.



Time Limit Reached

This phase uses the messages listed in the following table:

Messages in Time Limit Reached

Message	Protocol
EventRevoked	Interaction Management

Agent Pulls Interaction

This set of models illustrates the following scenario:

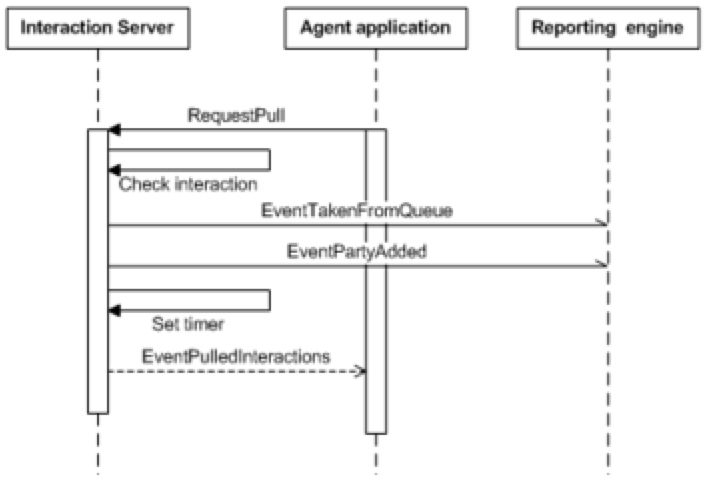
1. Agent application asks to pull an interaction:
   - From some place other than a workbin.
   - From a workbin.
2. Some processing activity occurs.
3. Timeout: Processing activity stops (or never occurred).

Agent Issues Pull Request

This phase has two versions, depending on whether the interaction is to be pulled from a workbin or from some other location.

From Non-Workbin

In this version of the first phase, shown in the following figure, the interaction is pulled from somewhere other than a workbin (most likely a queue). Notice that Interaction Server starts a timer, which continues running into the following phases.



Agent Pulls From Non-Workbin

This phase uses the messages listed in the following table:

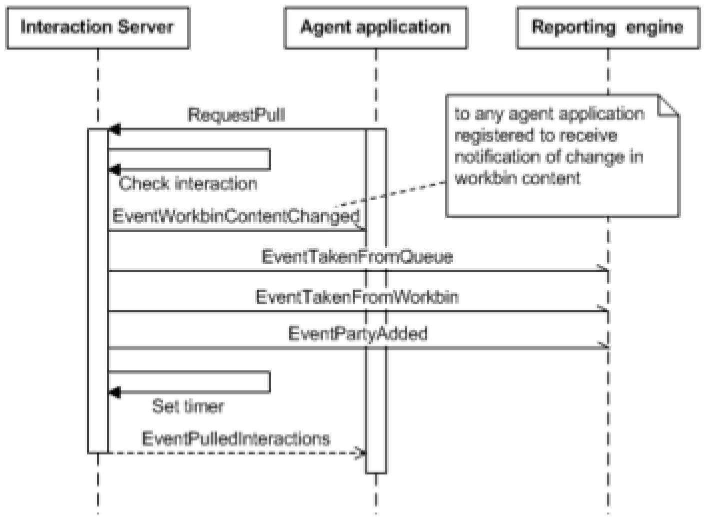
Messages in Agent Pulls From Non-Workbin

Message	Protocol
EventPartyAdded	Reporting

Message	Protocol
EventPulledInteractions	Interaction Management
EventTakenFromQueue	Reporting

From Workbin

In this version of the first phase, shown in the following figure, the interaction is pulled from a workbin.



Agent Pulls Interaction From Workbin

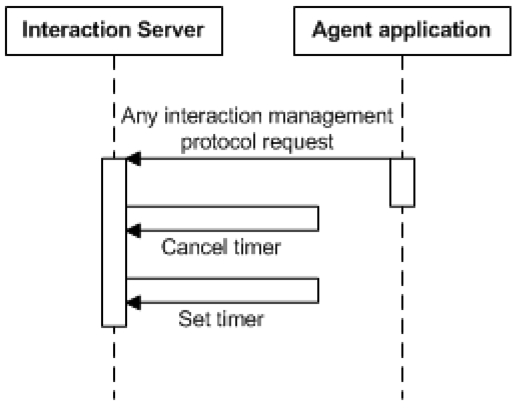
This version uses the same messages as the previously-described version, plus two more. All are listed in the following table:

Messages in Agent Pulls From Workbin

Message	Protocol
EventPartyAdded	Reporting
EventPulledInteractions	Interaction Management
EventTakenFromQueue	Reporting
EventTakenFromWorkbin	Reporting
EventWorkbinContentChanged	Interaction Management

Processing Occurs

In this phase, shown in the following figure, Interaction Server resets its timer each time it receives a request from the agent application. The interaction remains with the agent as long as the agent continues to send requests.

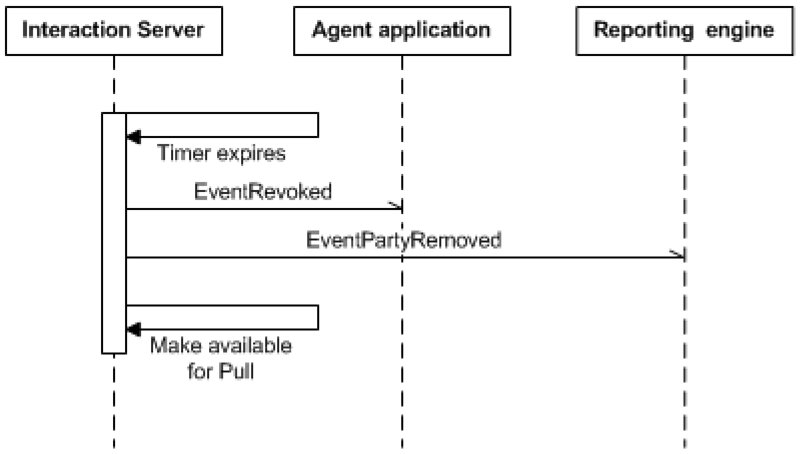


Processing Occurs

This phase uses any request from the Interaction Management Protocol.

No Processing: Timeout

In this phase, shown in the following figure, the timer expires and Interaction Server revokes the interaction.



No Processing: Timeout

This phase uses the messages listed in the following table:

Messages in No Processing: Timeout

Message	Protocol
EventPartyRemoved	Reporting
EventRevoked	Interaction Management

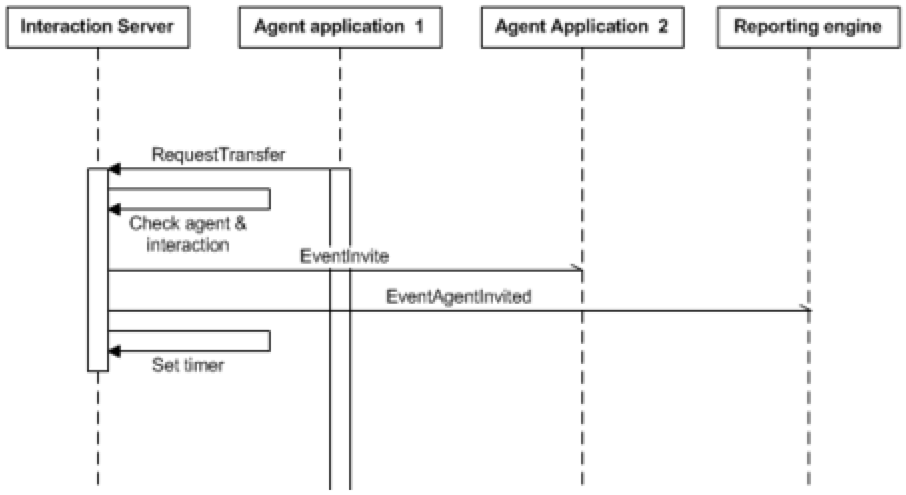
Transfer

This set of models illustrates the following scenario:

1. One agent invites another to accept a transfer.
2. One of the following happens:
    - The second agent accepts the invitation and the transfer completes.
    - The second agent rejects the invitation.
    - There is no response and the invitation times out.
    - Interaction Server finds that either the first agent application or the interaction is invalid.

Invitation Issued

In this phase, shown in the following figure, Agent 1 asks Interaction Server to invite Agent 2 to accept a transfer.



Transfer Invitation Issued

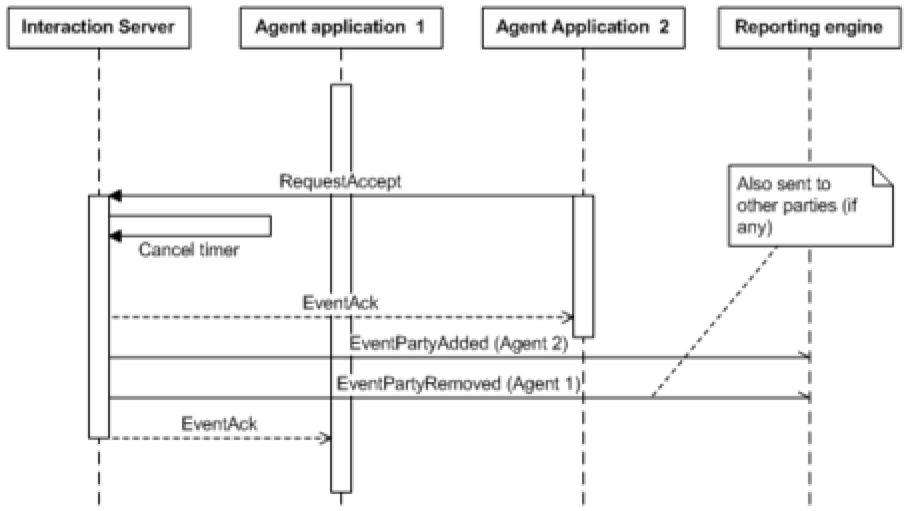
This phase uses the messages shown in listed in the following table:

Messages in Transfer Invitation Issued

Message	Protocol
EventAgentInvited	Reporting
EventInvite	Interaction Management

Invitation Accepted

In this phase, shown in the following figure, Agent 2 accepts the transfer.



Transfer Invitation Accepted

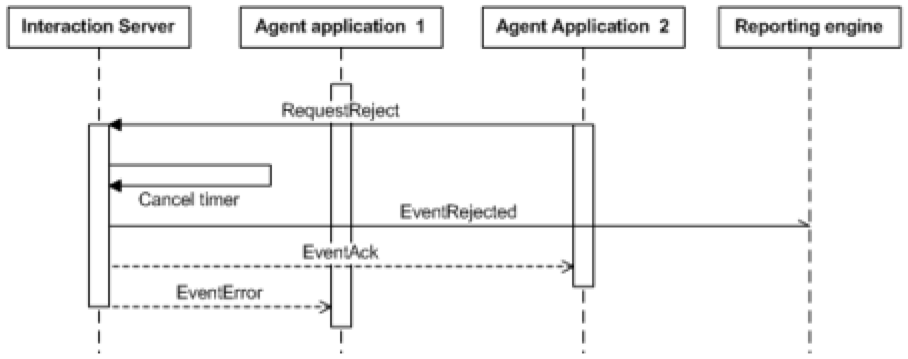
This phase uses the messages shown in the following table:

Messages in Transfer Invitation Accepted

Message	Protocol
EventAck	Interaction Management
EventPartyAdded	Reporting
EventPartyRemoved	Reporting

Invitation Rejected

In this phase, shown in the following figure, Agent 2 rejects the invitation.



Transfer Invitation Rejected

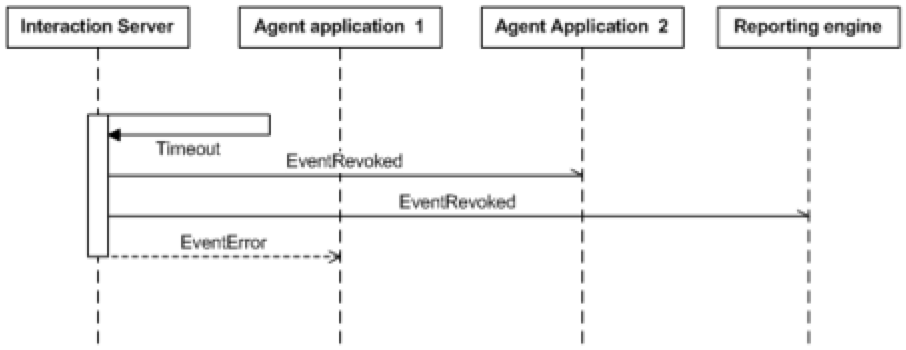
This phase uses the messages shown in the following table:

Messages in Transfer Invitation Rejected

Message	Protocol
EventAck	Interaction Management
EventError	Interaction Management
EventRejected	Reporting

Invitation Times Out

In this phase, shown in the following figure, Agent 2 does not respond within the timeout period, so Interaction Server revokes the invitation.



Transfer Invitation Times Out

This phase uses the messages shown in the following table:

Messages in Transfer Invitation Times Out

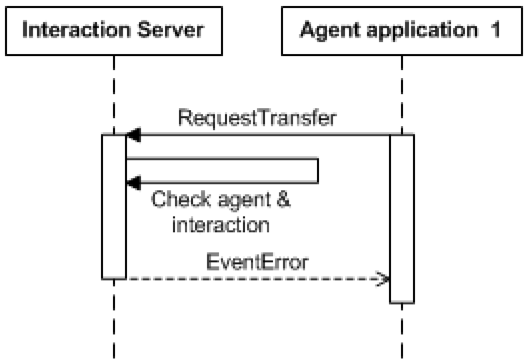
Message	Protocol
EventError	Interaction Management
EventRevoked	Interaction Management

Invitation Is Invalid

In this phase, shown in the following figure, Interaction Server finds that either the agent or the interaction is not valid (for example, the agent is not registered with Interaction Server).

Important

This phase replaces, rather than follows, the initial phase Invitation Issued.



Transfer Invitation Is Invalid

This phase uses the messages shown in the following table:

Messages in Transfer Invitation Is Invalid

Message	Protocol
EventError	Interaction Management

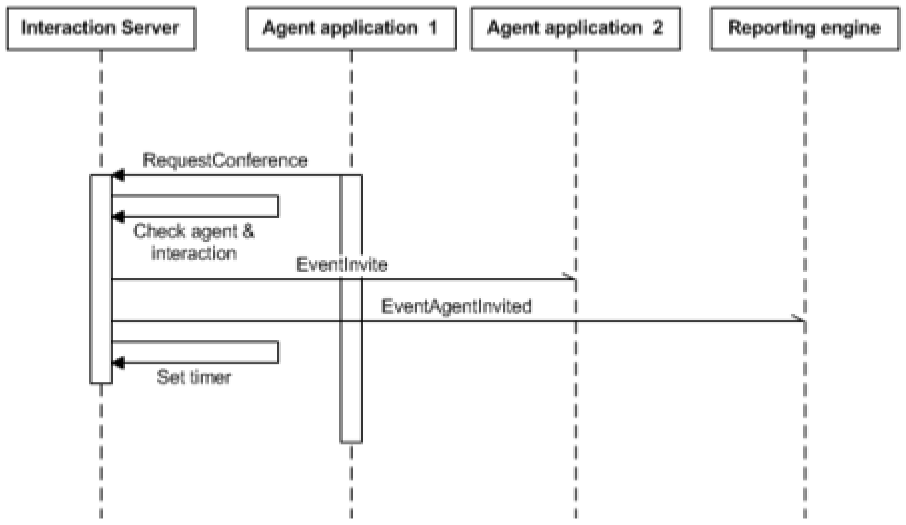
Conference

This set of models illustrates the following scenario:

1. One agent invites another to join a conference.
2. One of the following happens:
    - The second agent accepts the invitation and joins the conference.
    - The second agent rejects the invitation.
    - There is no response and the invitation times out.
    - Interaction Server finds that either the first agent application or the interaction is invalid.
3. If the second agent accepts, the conference proceeds until the second agent then attempts to leave the conference, successfully or not.

Invitation Issued

In this phase, shown in the following figure, Agent 1 invites Agent 2 to join a conference.



Conference Invitation Issued

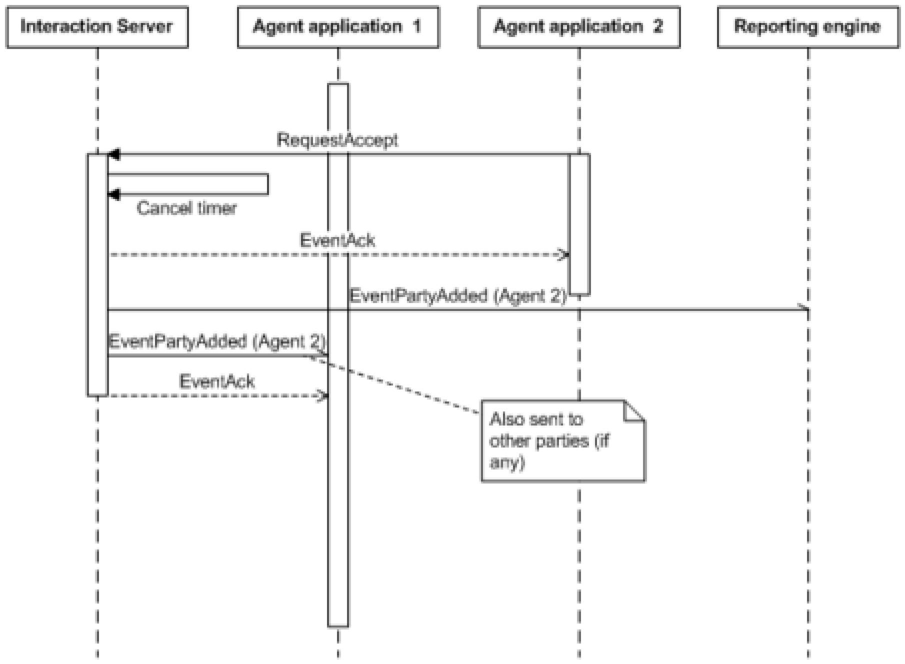
This phase uses the messages shown in the following table:

Messages in Conference Invitation Issued

Message	Protocol
EventAgentInvited	Reporting
EventInvite	Interaction Management

Invitation Accepted

In this phase, shown in the following figure, Agent 2 accepts the invitation and Interaction Server informs all parties to the interaction that Agent 2 has joined.



Conference Invitation Accepted

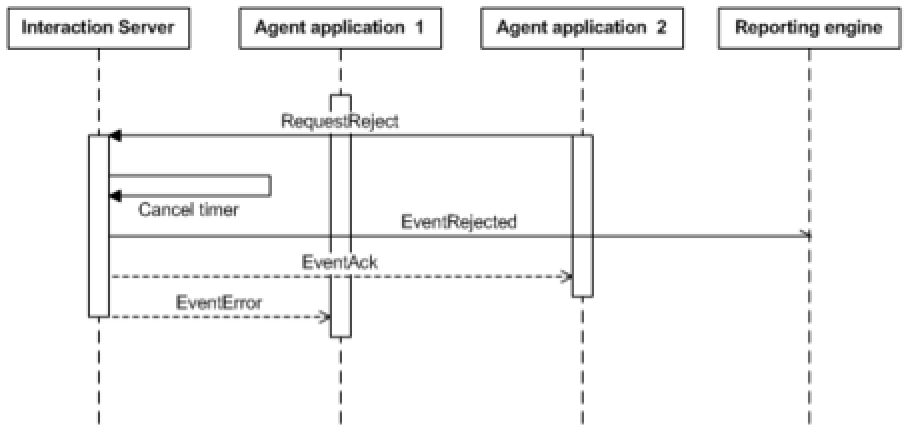
This phase uses the messages shown in the following table:

Messages in Conference Invitation Accepted

Message	Protocol
EventAck	Interaction Management
EventPartyAdded	Reporting and Interaction Management

Invitation Rejected

In this phase, shown in the following figure, Agent 2 rejects the invitation.



Conference Invitation Rejected

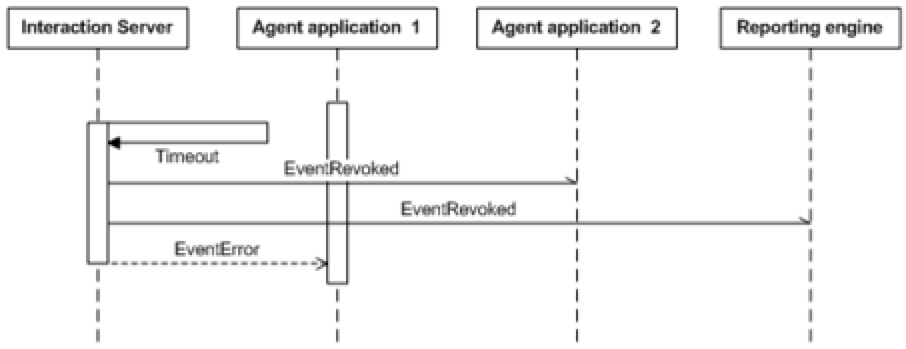
This phase uses the messages shown in the following table:

Messages in Conference Invitation Rejected

Message	Protocol
EventAck	Interaction Management
EventError	Interaction Management
EventRejected	Reporting

Invitation Times Out

In this phase, shown in the following figure, Agent 2 does not respond within the timeout period, so Interaction Server revokes the invitation.



Conference Invitation Times Out

This phase uses the messages shown in the following table:

Messages in Conference Invitation Time Out

Message	Protocol
EventError	Interaction Management

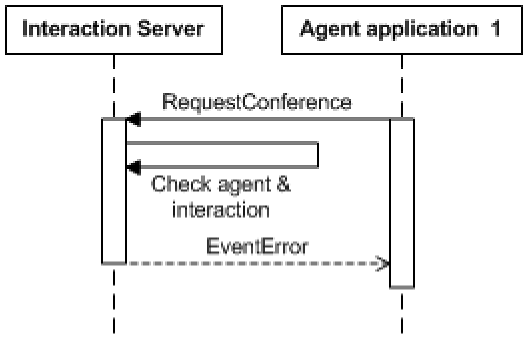
Message	Protocol
EventRevoked	Interaction Management

Invitation Is Invalid

In this phase, shown in the following figure, Interaction Server finds that either the agent or the interaction is not valid (for example, the agent is not registered with Interaction Server).

Important

This phase replaces, rather than follows, the initial phase Invitation Issued.



Conference Invitation Is Invalid

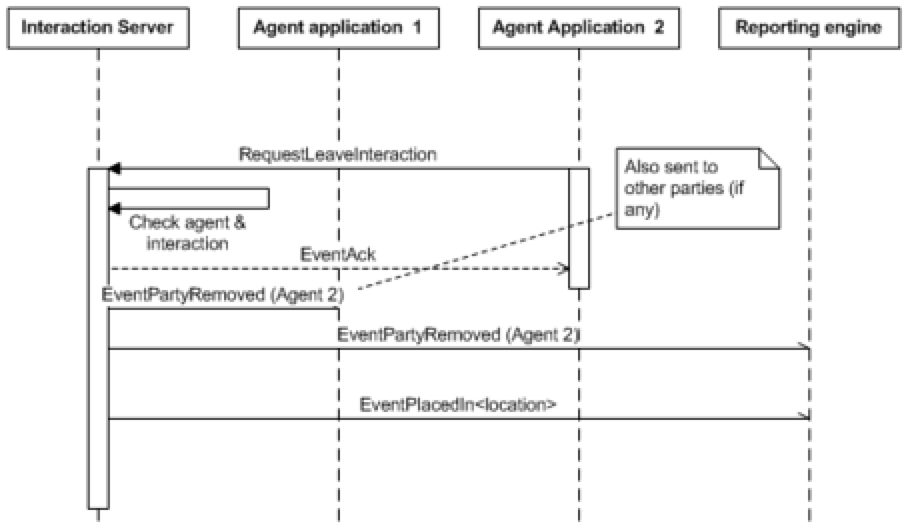
This phase uses the messages shown in the following table:

Messages in Conference Invitation Is Invalid

Message	Protocol
EventError	Interaction Management

Leave the Conference

In this phase, shown in the following figure, Agent 2 leaves the conference.



Leave the Conference

When the last party leaves the interaction, Interaction Server returns the interaction to its former location and informs the Reporting engine.

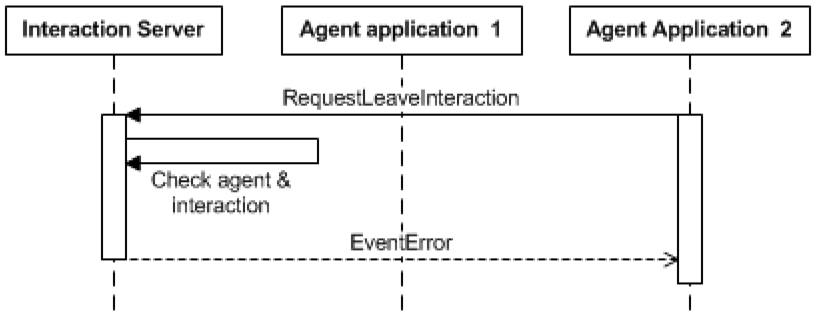
This phase uses the messages shown in the following table:

Messages in Leave the Conference

Message	Protocol
EventAck	Interaction Management
EventPartyRemoved	Interaction Management
EventPartyRemoved	Reporting
EventPlacedInQueue	Reporting
EventPlacedInWorkbin	Reporting

Fail to Leave the Conference

In this phase, shown in the following figure, Agent 2 attempts to leave the conference, but Interaction Server rejects the request.



Fail to Leave the Conference

This phase uses the messages shown in the following table:

Messages in Fail to Leave the Conference

Message	Protocol
EventError	Interaction Management

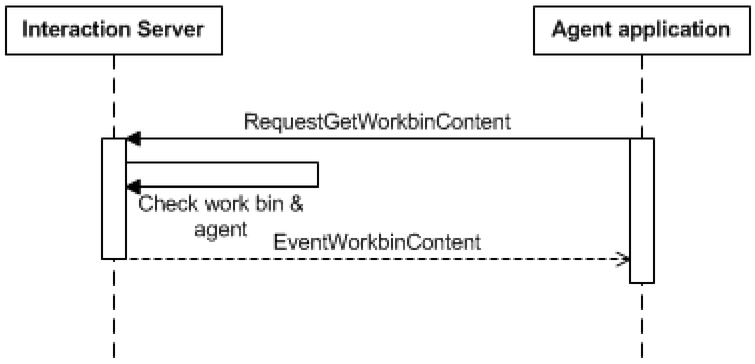
Workbin Operations

This set of models illustrates the following two ways that an agent application can interact with a workbin:

- Get content from the workbin
- Register to receive notification when the workbin's content changes.

Agent Gets Workbin Content

In this phase, shown in the figure below, the agent application requests and receives data on the interactions that the workbin contains.



Agent Gets Workbin Content

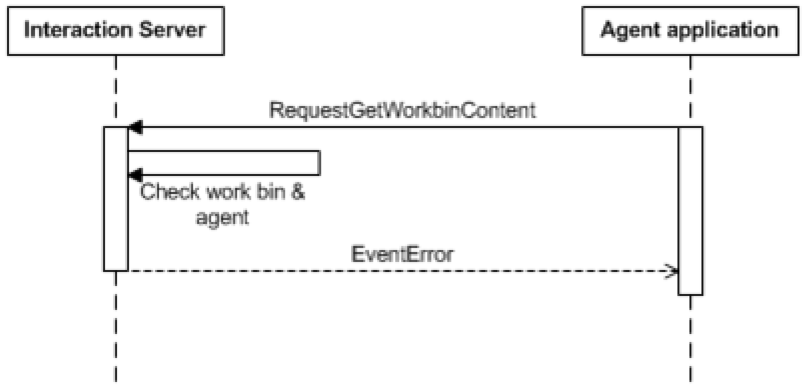
This phase uses the messages shown in the following table:

Messages in Agent Gets Workbin Content

Message	Protocol
EventWorkbinContent	Interaction Management

Agent Fails to Get Workbin Content

In this phase, shown in the figure below, Interaction Server rejects the agent's request for workbin data. This happens when either the agent or the workbin is not registered with Interaction Server.



Agent Fails to Get Workbin Content

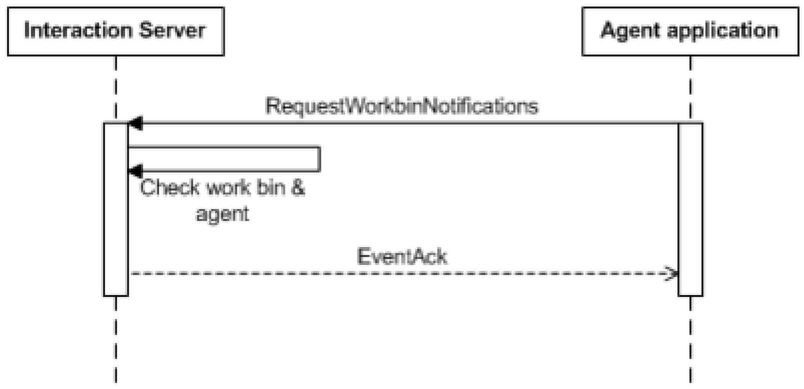
This phase uses the messages shown in the following table:

Messages in Agent Fails to Get Workbin Content

Message	Protocol
EventError	Interaction Management

Agent Requests Workbin Notification

In this phase, shown in the the figure below, the agent asks to be notified of all future changes in the contents of a specified workbin.



Agent Requests Workbin Notification

Interaction Server checks that both the workbin and the agent are registered with it.

- If either is unknown to Interaction Server, it returns `EventError`.
- If both are registered, Interaction returns `EventAck` to the agent.

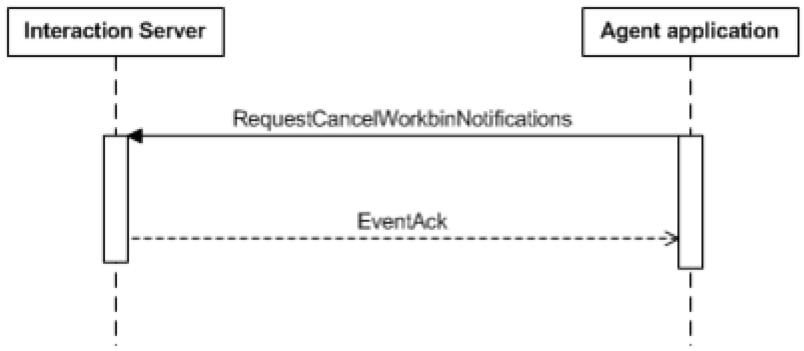
This phase uses the messages shown in the following table:

Messages in Agent Requests Workbin Notification

Message	Protocol
EventAck	Interaction Management

Agent Cancels Workbin Notifications

In this phase, shown in the figure below, the agent cancels its subscription for workbin notification.



Agent Cancels Workbin Notifications

If either the workbin or the agent is unknown to Interaction Server, or if the agent has not subscribed for workbin notification, Interaction Server returns `EventError`.

This phase uses the messages shown in the following table:

Messages in Agent Cancels Workbin Notifications

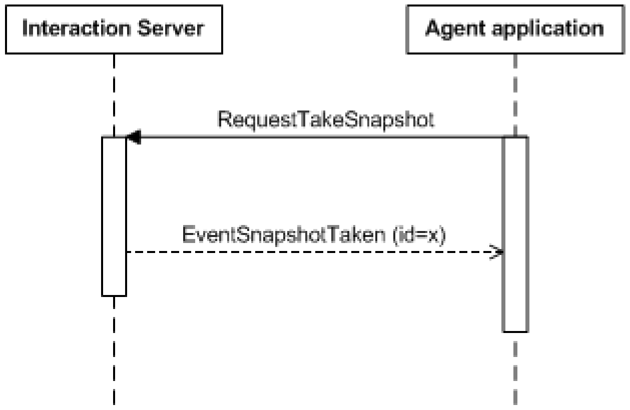
Message	Protocol
EventAck	Interaction Management

Snapshot Operations

This set of models illustrates various operations on snapshots. A snapshot is a list of all of the interactions in the Interaction Server's database that meet specified conditions at a given time. The agent application requests a snapshot from Interaction Server and uses the results to populate the list of interactions that display on the Agent or Supervisor desktop.

Take a Snapshot

In this phase, shown in the figure below, the agent application defines a set of conditions and requests a snapshot of the interactions that meet the conditions.



Take a Snapshot

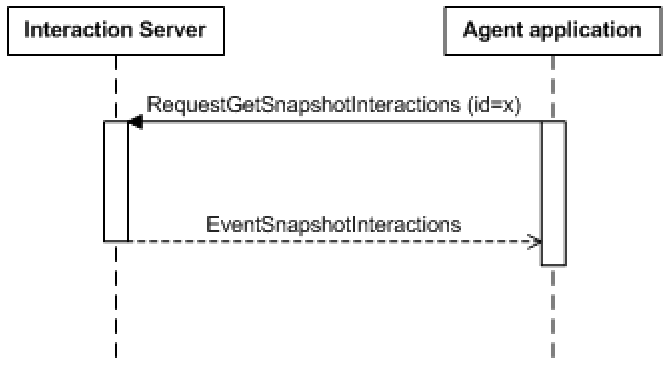
This phase uses the messages shown in the following table:

Messages in Take a Snapshot

Message	Protocol
EventSnapshotTaken	Interaction Management

Get Snapshot Interactions

In this phase, shown in the following figure, the agent application requests the content of a previously taken snapshot; that is, information about the interactions that are included in the snapshot.


Interaction Server
Agent application
RequestGetSnapshotInteractions (id=x)
EventSnapshotInteractions

Get Snapshot Interactions

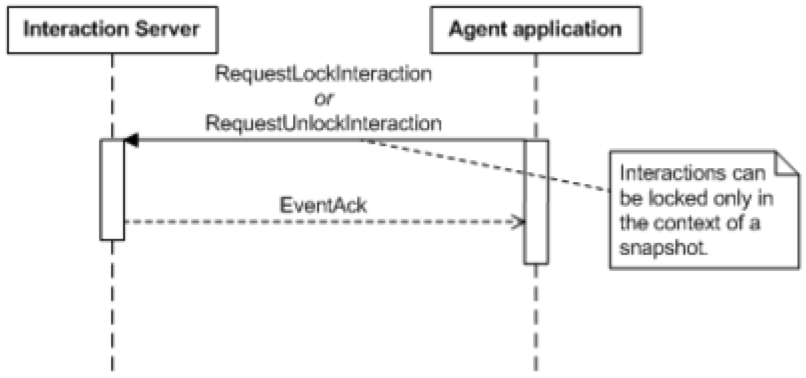
This phase uses the messages shown in the following table:

Messages in Get Snapshot Interactions

Message	Protocol
EventSnaphotInteractions	Interaction Management

Lock or Unlock Interactions

In this phase, shown in the figure below, an agent application asks Interaction Server to lock or unlock an interaction. Locked interactions are not visible to views and can not be pulled from queues by URS or an agent.



Lock or Unlock Interactions

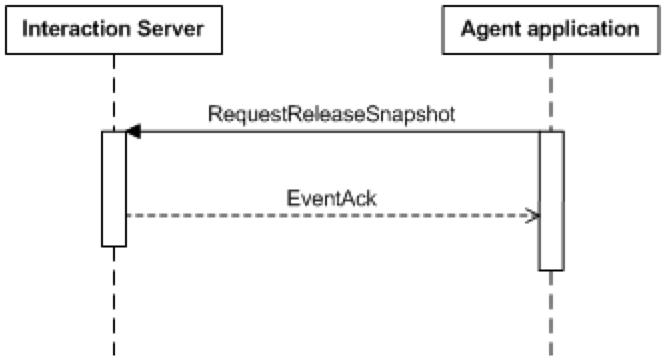
This phase uses the messages shown in the following table:

Messages in Lock or Unlock Interactions

Message	Protocol
EventAck	Interaction Management

Release Snapshot

In this phase, shown in the figure below, an agent application asks Interaction Server to release a specified snapshot; that is, to unlock any interactions that were locked in the context of this snapshot.



Release Snapshot

This phase uses the messages shown in the following table:

Messages in Release Snapshot

Message	Protocol
EventAck	Interaction Management

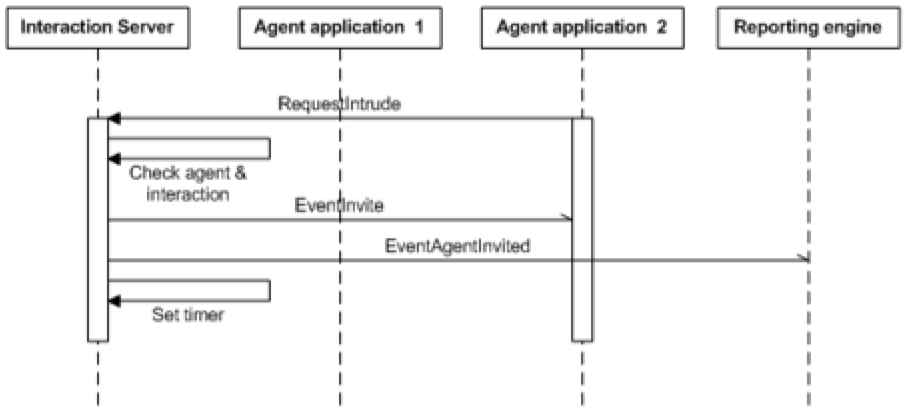
Intrusion

Intrusion is like a conference, except that a conference is initiated by an entity that is already a party to the interaction, while intrusion is initiated by an entity that is not a party to the interaction. Therefore intrusion may also be described as an externally-initiated conference. This set of models illustrates the following scenario:

1. While Agent 1 is processing an interaction, Agent 2 asks to join in a conference.
2. One of the following happens:
    - The conference is set up and proceeds.
    - Agent 2 declines the conference.
    - The request times out.
    - Interaction Server rejects Agent 2's request.

Intrusion Requested

In this phase, shown in the figure below, Agent 2 asks to join an interaction that is already being processed by Agent 1. Interaction Server responds by sending `EventInvite` to Agent 2.



Intrusion Requested

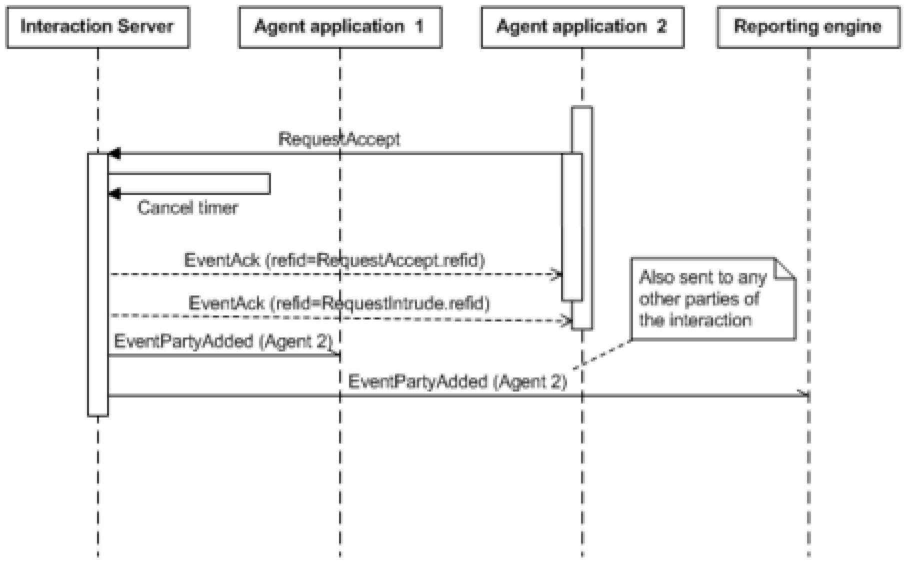
This phase uses the messages shown in the following table:

Messages in Intrusion Requested

Message	Protocol
EventAgentInvited	Reporting
EventInvite	Interaction Management

Intrusion Accepted

In this phase, shown in the figure below, Agent 2 accepts the invitation to join the interaction. Interaction Server responds with two instances of `EventAck`: the first one acknowledges the agent's `RequestAccept` from this phase, the other acknowledges the agent's `RequestIntrude` from the preceding phase.


Interaction Server
Agent application 1
Agent application 2
Reporting engine
RequestAccept
Cancel timer
EventAck (refid=RequestAccept.refid)
EventAck (refid=RequestIntrude.refid)
EventPartyAdded (Agent 2)
EventPartyAdded (Agent 2)
Also sent to any other parties of the interaction

Intrusion Accepted

Interaction Server then reports the addition of Agent 2 to the interaction by sending `EventPartyAdded` to Agent 1, the reporting engine, and any other parties to the interaction.

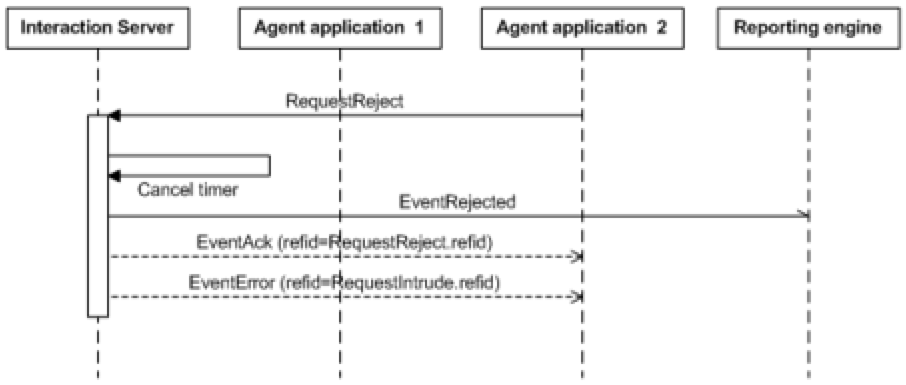
This phase uses the messages shown in the following table:

Messages in Intrusion Accepted

Message	Protocol
EventAck	Interaction Management
EventPartyAdded	Reporting and Interaction Management

Intrusion Rejected by Agent

In this phase, shown in the figure below, Agent 2 rejects the invitation to join the interaction. Interaction Server responds with `EventAck`, replying to `RequestReject` from this phase, and with `EventError`, replying to `RequestIntrude` from the previous Intrusion Requested.



Intrusion Rejected by Agent

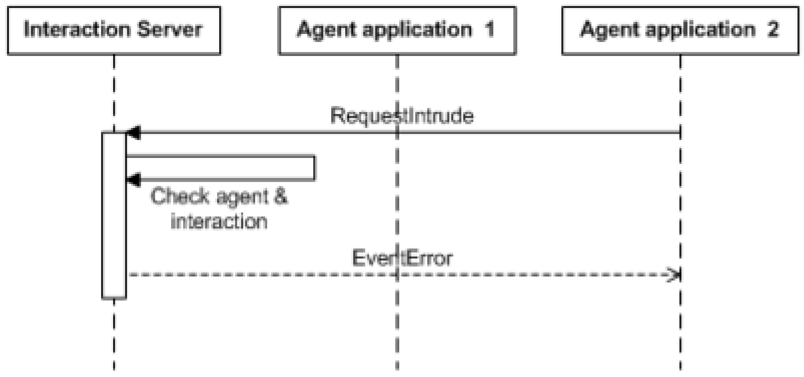
This phase uses the messages shown in the following table:

Messages in Rejected by Agent

Message	Protocol
EventAck	Interaction Management
EventError	Interaction Management
EventRejected	Reporting

Intrusion Rejected by Interaction Server

In this phase, shown in the figure below, Interaction Server finds that either the agent or the interaction are not registered, and so rejects Agent 2's request for intrusion.



Intrusion Rejected by Interaction Server

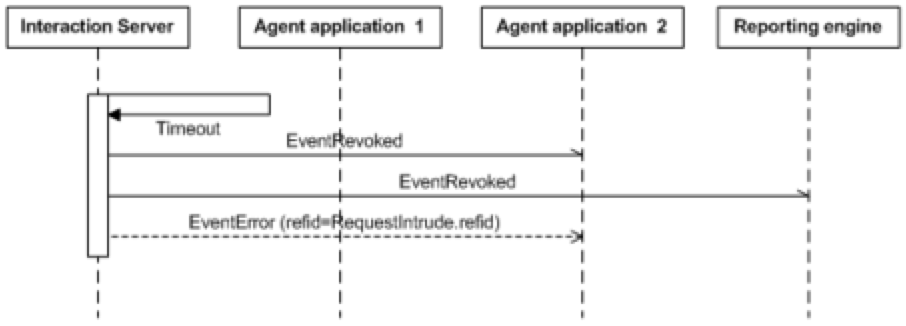
This phase uses the messages shown in the following table:

Messages in Intrusion Rejected by Interaction Server

Message	Protocol
EventError	Interaction Management

Intrusion Times Out

In this phase, shown in the figure below, the timer that Interaction Server started in the first phase (Intrusion Requested) expires. Interaction Server then sends `EventRevoked` to Agent 2 and to the reporting engine. It also sends `EventError` as a response to Agent 2's original `RequestIntrude` in the first phase.



Intrusion Times Out

This phase uses the messages shown in the following table:

Messages in Intrusion Times Out

Message	Protocol
EventError	Interaction Management
EventRevoked	Interaction Management

Login/Logout

This set of models illustrates a scenario which follows Successful Registration:

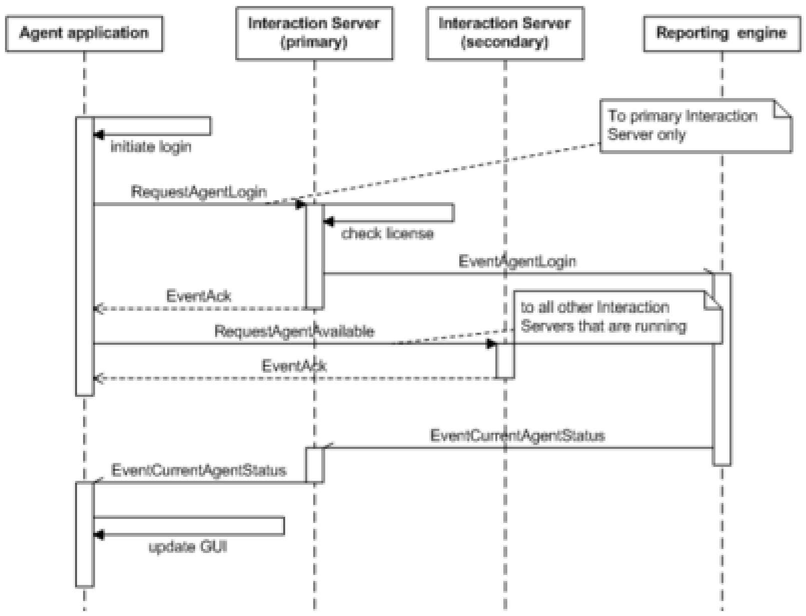
1. After connecting and registering, the agent application logs in to all available Interaction Servers.
2. The agent application logs out from the Interaction Servers.

Agent Logs In

In this phase, shown in the figure below:

1. The agent application sends `RequestAgentLogin` to the primary Interaction Server.
2. The primary Interaction Server sends `EventAgentLogin` to the reporting engine, which responds with `EventCurrentAgentStatus`.
3. The agent applications sends `RequestAgentAvailable` to all other Interaction Servers that are running.
4. The primary Interaction Server relays `EventCurrentAgentStatus` to the agent application.

Note that the agent application uses different events to log in to primary versus secondary Interaction Servers.



Agent Logs In

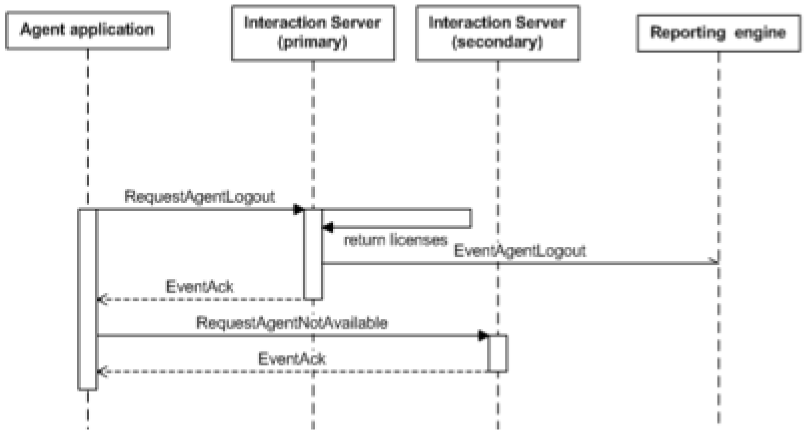
This phase uses the messages shown in the following table:

Messages in Agent Logs In

Message	Protocol
EventAck	Interaction Management
EventAgentLogin	Reporting
EventCurrentAgentStatus	Reporting

Agent Logs Out

In this phase, shown in the figure below, the agent application sends `RequestAgentLogout` to the primary Interaction Server, which relays that information to the reporting engine. After the agent application receives `EventAck` from this Interaction Server, it sends `RequestAgentNotAvailable` to all secondary Interaction Servers.



Agent Logs Out

This phase uses the messages shown in the following table:

Messages in Agent Logs Out

Message	Protocol
EventAck	Interaction Management
EventAgentLogout	Reporting

Reporting Engine Connects

This model illustrates the following scenario:

1. The reporting engine connects with Interaction Server, then uses `RequestStartPlaceAgentReportingAll` to request data on all agents that are logged in to this Interaction Server.
2. Interaction Server uses `EventPlaceAgentState` to inform the reporting engine of the state (media state, interactions being handled) of all agents that are logged in with this Interaction Server.
3. The reporting engine (in this case, Stat Server) combines this state information with any applicable capacity rules to calculate the agent's status. It relays the status information to Interaction Server using `EventCurrentAgentStatus.` Interaction Server passes the same event to the agent application, which displays the status information in its UI.

   Important

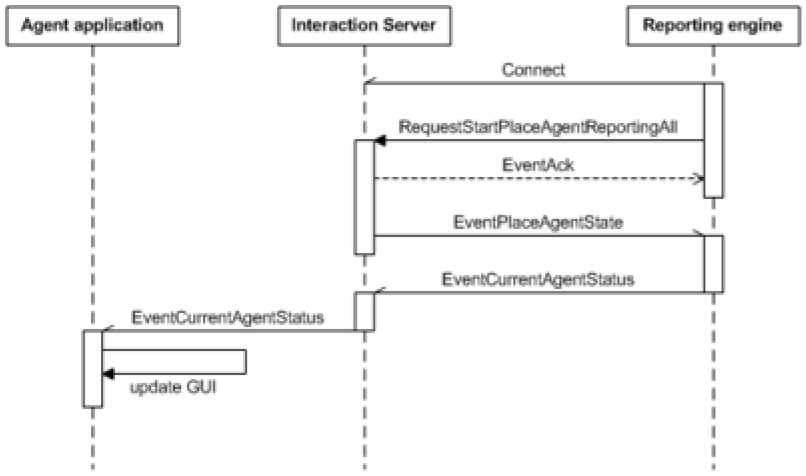
   In this scenario the reporting engine must be Stat Server. In the current release, Stat Server is the only component that calculates capacity.

4. If there are multiple Interaction Servers, the reporting engine then connects with each one in turn and repeats Steps 2 and 3 with each.

Note that this model distinguishes between *state* and *status* of an agent, as follows:

- State indicates the media that this agent is ready to use and the interactions that the agent is currently handling.
- Status is the output of an Agent Capacity Rule, which Stat Server calculates using the agent's state as part of the input.

This model, which is not further divided into phases, is shown in the following figure:



Reporting Engine Connects

This model uses the messages shown in the following table:

Messages in Reporting Engine Connects

Message	Protocol
EventAck	Interaction Management
EventCurrentAgentStatus	Reporting
EventPlaceAgentState	Reporting

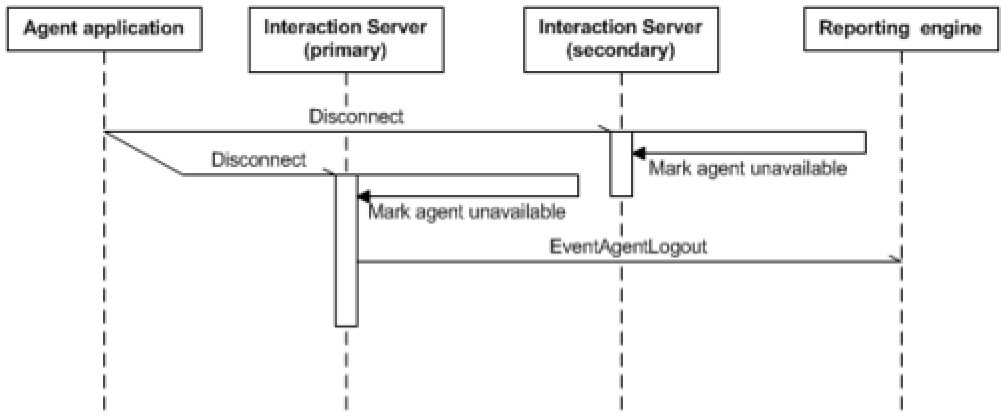
Disconnection and Failover

This set of models illustrates the following scenarios:

- The agent application disconnects from Interaction Server(s).
- The Interaction Server disconnects from agent application and reporting engine. Then the following happens:
    - The agent application connects to secondary Interaction Server, which becomes primary.
    - The former primary Interaction Server restarts.

Agent Disconnects

In this phase, shown in the figure below, the agent application disconnects from Interaction Server. This may due to a failure or to normal shutdown.



Agent Disconnects

The primary Interaction Server sends `EventAgentLogout` only if the agent was known to be logged in.

This phase uses the message shown in the following table:

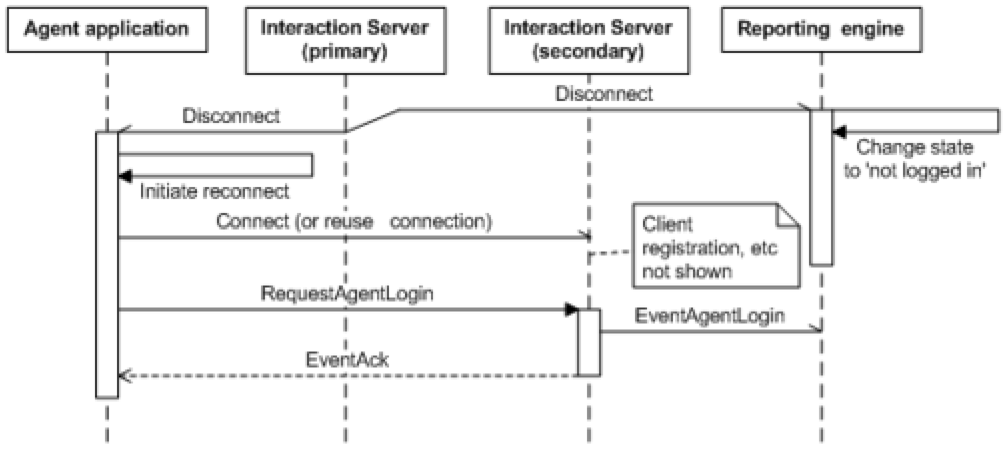
Messages in Agent Disconnects

Message	Protocol
EventAgentLogout	Reporting

Interaction Server Disconnects

In this phase, shown in the figure below:

1. The primary Interaction Server disconnects from the agent application and the reporting engine. This may due to a failure or to normal shutdown. The reporting engine sets the state of each agent logged in with this Interaction Server to `not logged in`.
2. The agent application connects to the secondary Interaction Server and registers with it. Registration is not shown here; see Registration for a description.
3. The agent application logs in to the secondary Interaction Server.
4. The secondary Interaction Server responds with `EventAck`, thereby becoming the new primary Interaction Server.


Agent application
Interaction Server (primary)
Interaction Server (secondary)
Reporting engine
Disconnect
Disconnect
Change state to 'not logged in'
Initiate reconnect
Connect (or reuse connection)
Client registration, etc not shown
RequestAgentLogin
EventAgentLogin
EventAck

Interaction Server Disconnects

This phase uses the messages shown in the following table:

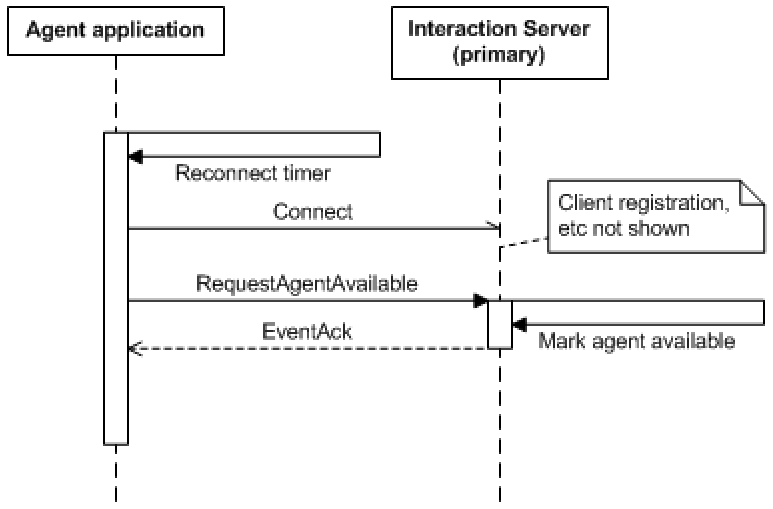
Messages in Interaction Server Disconnects

Message	Protocol
EventAck	Interaction Management
EventAgentLogin	Reporting

Interaction Server Restarts

In this phase, shown in the figure below:

1. The original primary Interaction Server restarts.
2. The agent application connects to it and registers (registration is not shown here; see Registration).
3. The agent application logs in to the Interaction Server using `RequestAgentAvailable`, upon which this Interaction Server becomes secondary.



Interaction Server Restarts

This phase uses the messages shown in the following table:

Messages in Interaction Server Restarts

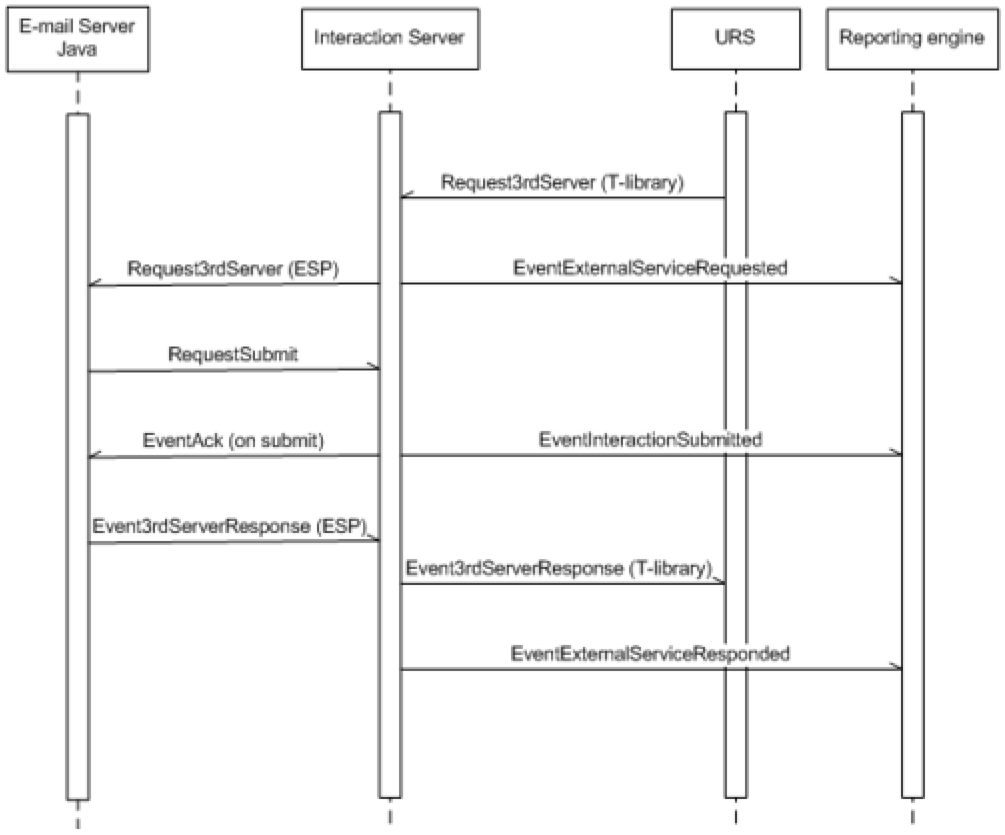
Message	Protocol
EventAck	Interaction Management

Invoke Autoresponse

This set of models illustrates the following scenario:

1. URS, triggered by an Autoresponse strategy object, generates a request for E-mail Server Java to generate a new autoresponse interaction. It sends this request to Interaction Server using `Request3rdServer`.
2. Interaction Server relays the request to E-mail Server Java using the same `Request3rdServer.` It also relays the content of the request to the reporting engine using `EventExternalServiceRequested.`
3. E-mail Server Java generates a new Autoresponse interaction and submits it to Interaction Server using `RequestSubmit`. When Interaction Server acknowledges the submission, E-mail Server Java sends `Event3rdServerResponse`, which Interaction Server relays to URS. Interaction Server also relays the content of that event to the reporting engine using `EventExternalServiceResponded.`

This model, which is not further divided into phases, is shown below.


E-mail Server Java
Interaction Server
URS
Reporting engine
Request3rdServer (T-library)
Request3rdServer (ESP)
EventExternalServiceRequested
RequestSubmit
EventAck (on submit)
EventInteractionSubmitted
Event3rdServerResponse (ESP)
Event3rdServerResponse (T-library)
EventExternalServiceResponded

Invoke Autoresponse

This phase uses the messages shown in the following table:

Messages in Invoke Autoresponse

Message	Protocol
Event3rdServerResponse	ESP
EventAck	Interaction Management
EventExternalServiceRequested	Reporting
EventExternalServiceResponded	Reporting
EventInteractionSubmitted	Reporting

In this scenario, URS uses Interaction Server as an intermediary to communicate with E-mail Server Java, which in this case is called a *third-party server.* Another example of such a third-party server is Classification Server, with which URS must similarly communicate when a strategy includes a Classify object. To relay these messages to third-party servers, Interaction Server uses the T-Library messages `RequestPrivateService` and `EventPrivateInfo`, with special content in their `extensions` and `user_data` attributes. With that special content, these messages make up a small *External Services Protocol* (ESP), as shown in the following table:

ESP Messages

T-Library Message	ESP Message	Description
EventPrivateInfo	Event3rdServerResponse	Returns results of third-party server's operation
	Event3rdServerFault	Contains information about failure of third-party server's operation

ESP is not further described in this manual. However, much of the relevant content of `Request3rdServer` and `Event3rdServerResponse` is repeated in the attributes of the reporting protocol events `EventExternalServiceRequested` and `EventExternalServiceResponded`.

Components that understand ESP are called *ESP servers.* Classification Server is an ESP server. E-mail Server Java functions both as a media server (processing Interaction Management Protocol messages) and as an ESP server, processing ESP messages. You can also create custom ESP servers using the Genesys Open Media Platform SDK.

IVR Call Flows

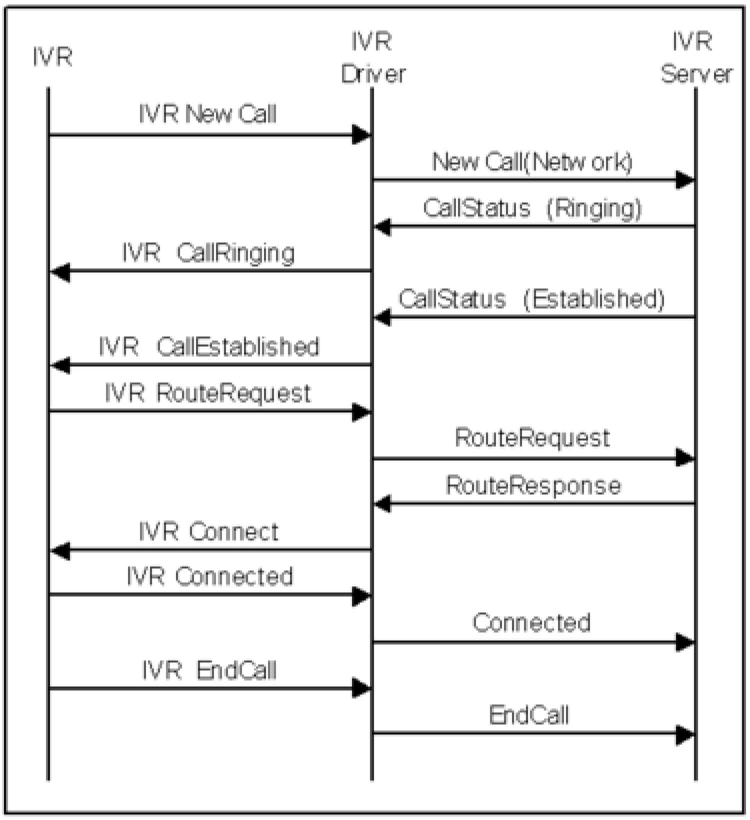
This section includes call flow diagrams that show all of the commonly encountered request-response sequences needed to create your IVR driver client.

This section contains the following subsections:

- Routing Call Flow
- Call Treatment Call Flows
- MakeCall Call Flow
- Conference Call Flow Diagrams
- Transfer Call Flow Diagrams

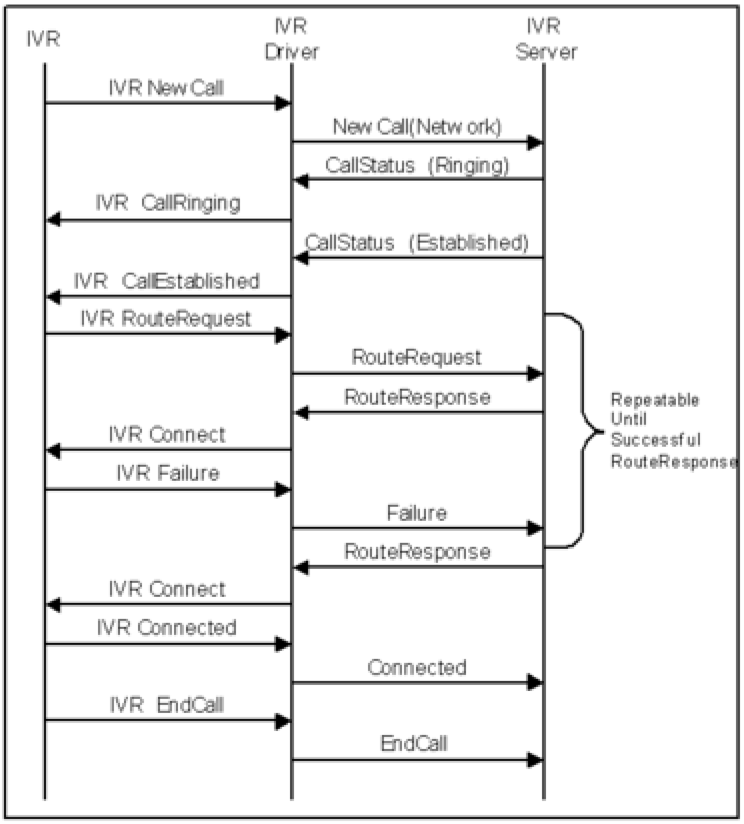
Routing Call Flow

Call Routing



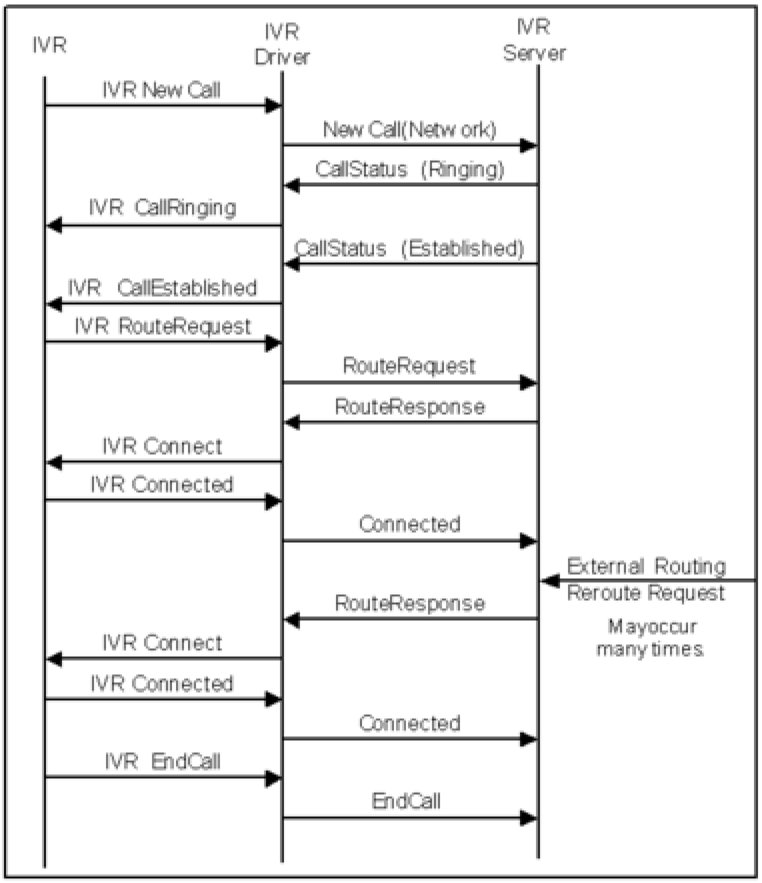
Call Routing Call Flow

Route Failed



Call Flow Showing Failed Route Attempt

Reroute

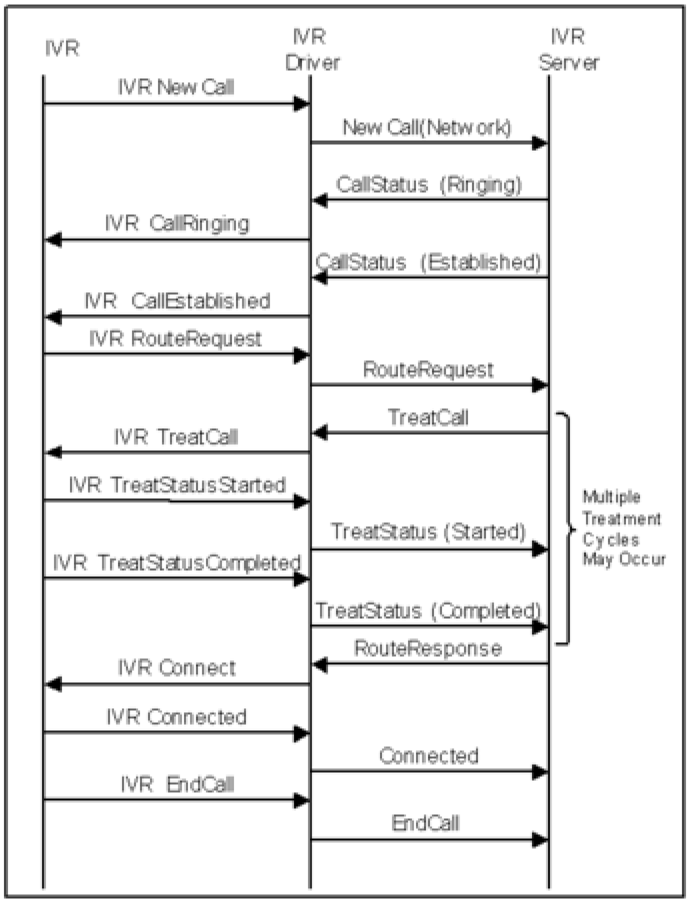


Rerouted Call

The external routing request is delivered from URS by the IVR Server.

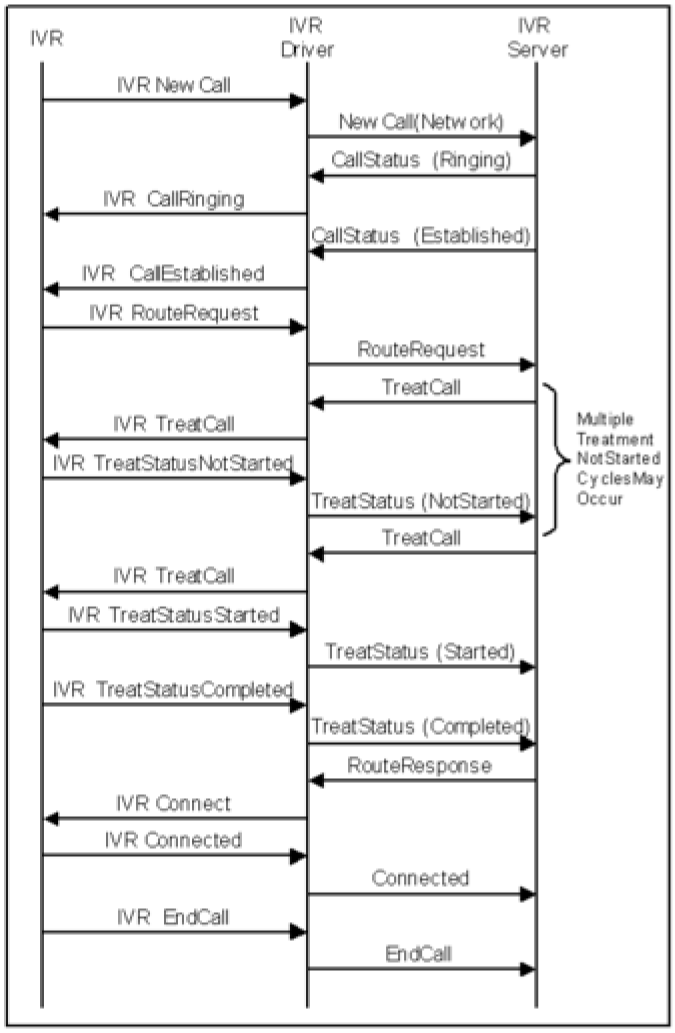
Call Treatment Call Flows

Call Treatment



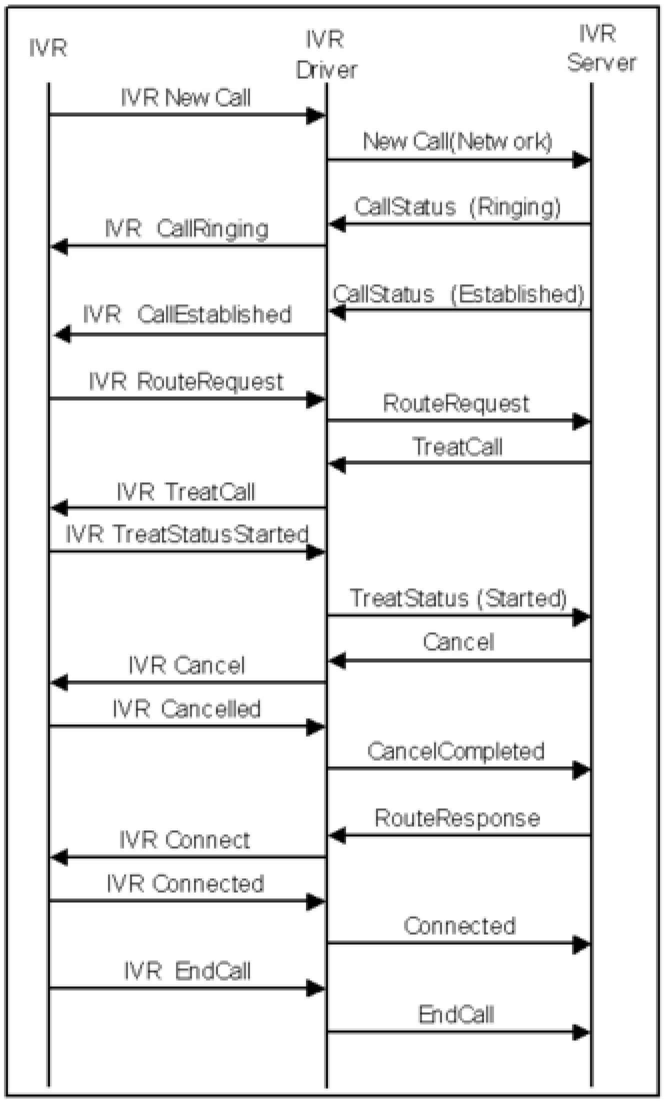
Call Treatment Call Flow

Call Treatment Failed



Call Treatment Failed Call Flow

Call Treatment Interrupted

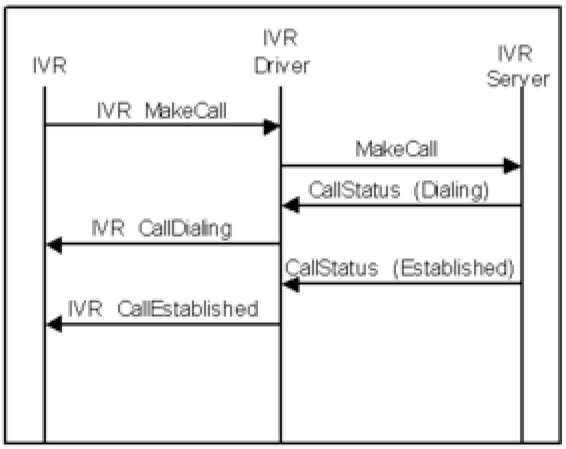


Interrupted Call Treatment Flow

The command to cancel the call treatment is forwarded from the Genesys Framework by IVR Server.

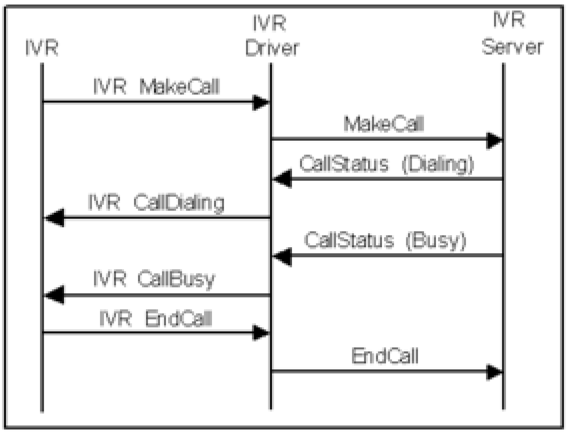
MakeCall Call Flow

MakeCall



MakeCall Operation

MakeCall (Busy)

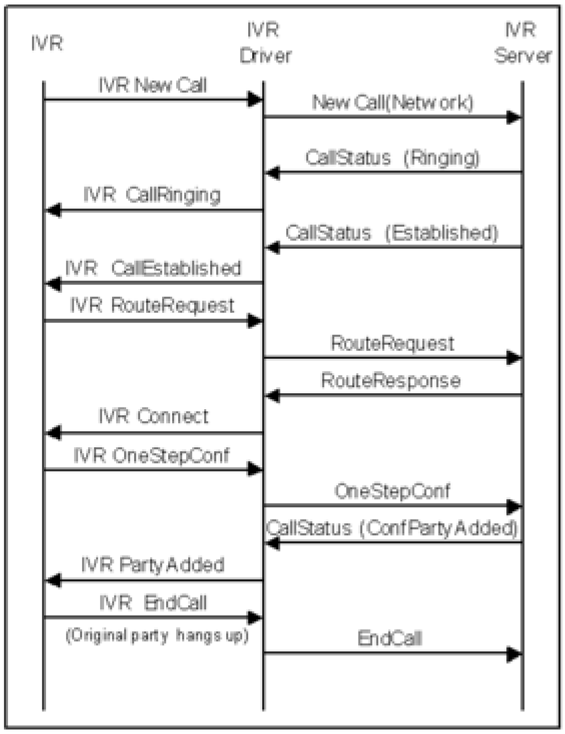


MakeCall (Busy) Call Flow

Conference Call Flow Diagrams

The following call flow diagrams illustrate several scenarios involving conferenced calls.

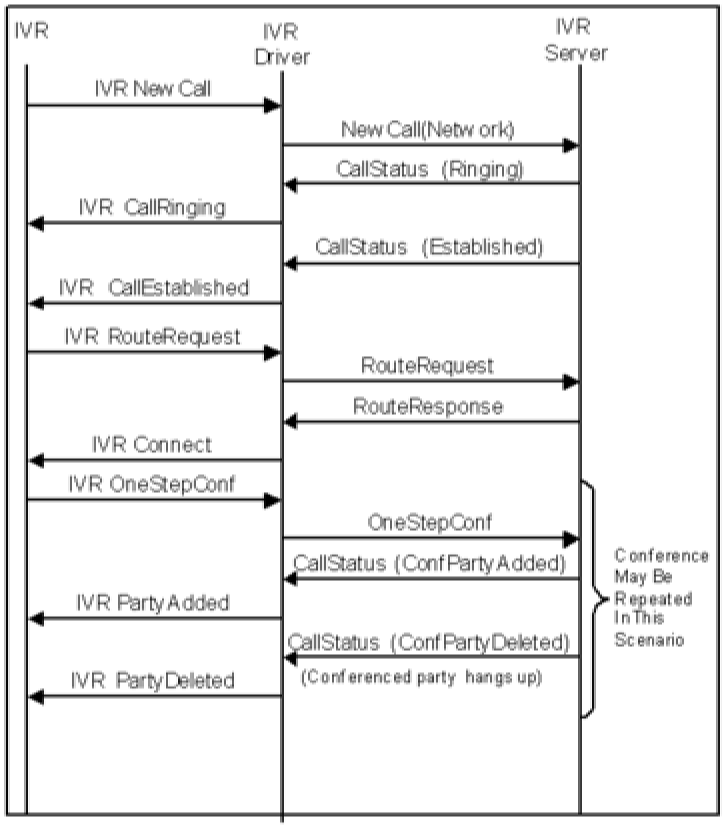
One-Step Conference



Call Flow for a One-Step Conference

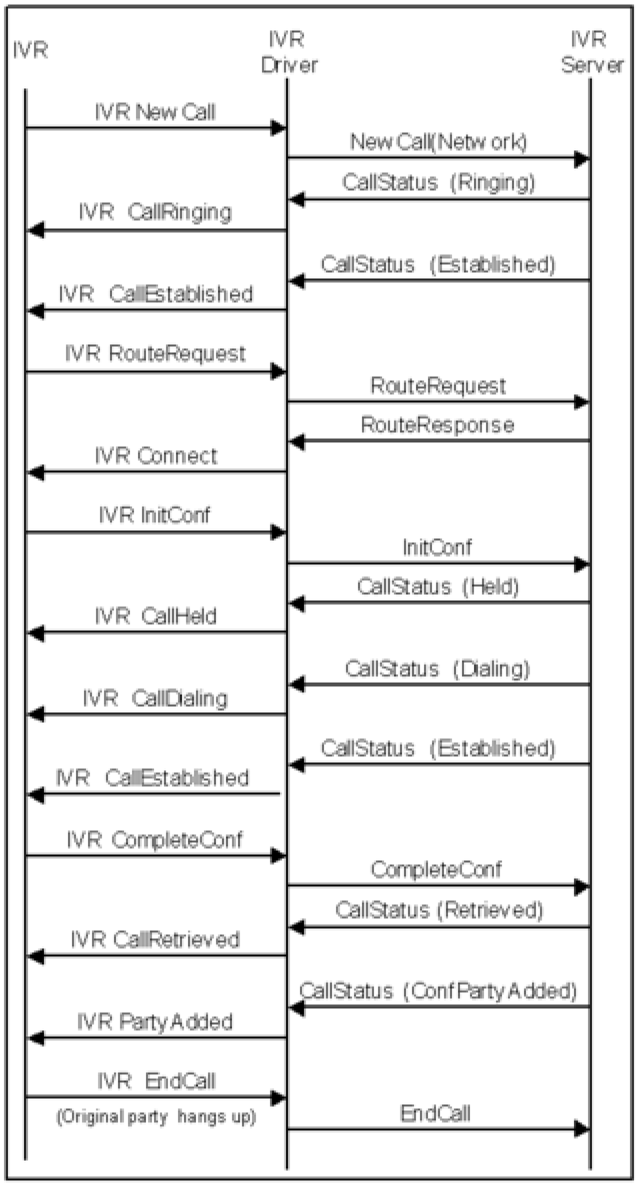
If a `CallError` occurs, IVR Server automatically returns you to the same status as before the conference call was started. This means that the original call is retrieved without any input from the IVR.

One-Step Conference, Scenario 2



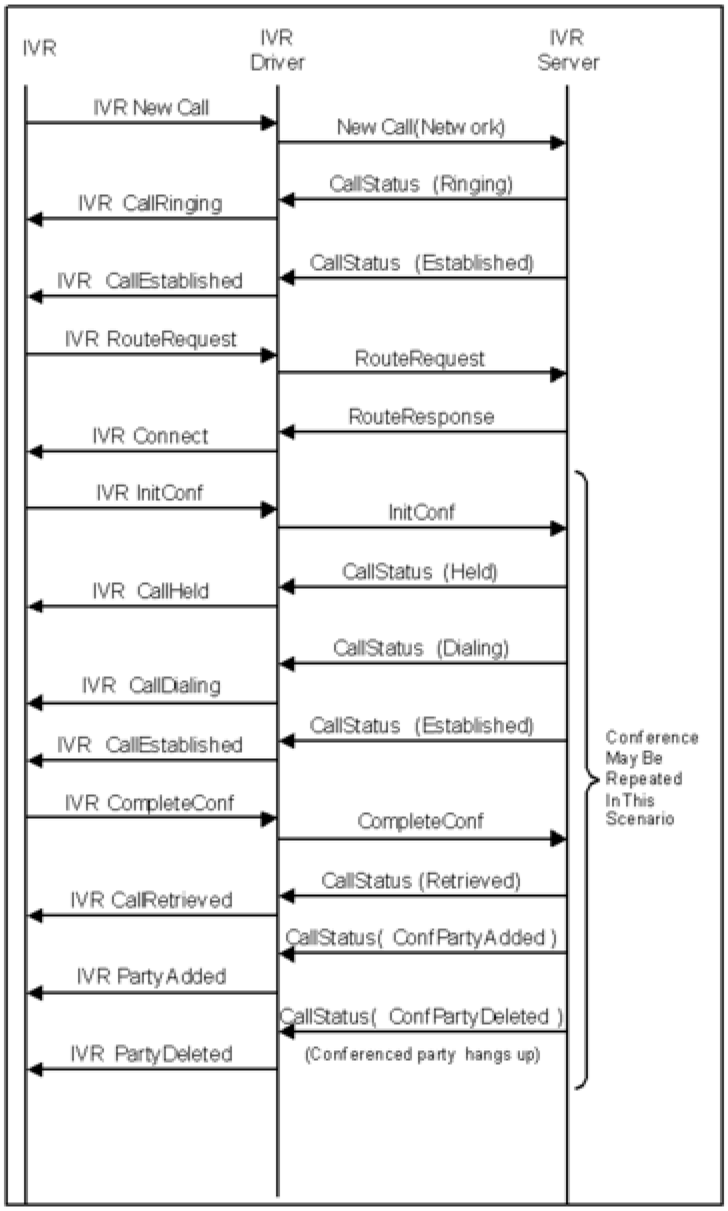
One-Step Conference with Alternative Disconnect Scenario

Conference Consult Call



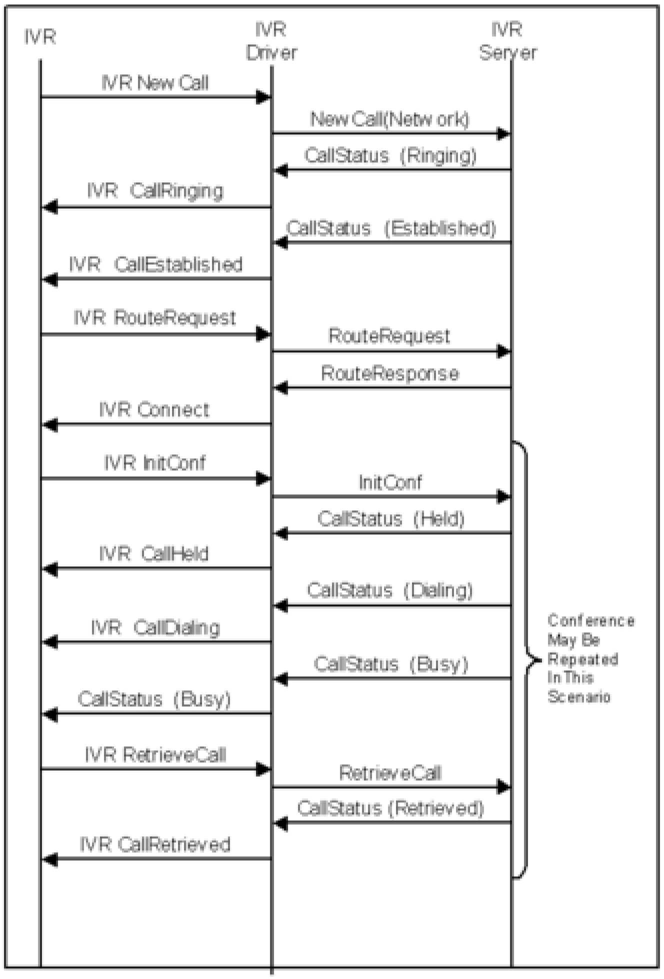
Call Flow for Conference Consult Call

Conference Consult Call, Scenario 2



Conference Consult Call Alternative Scenario

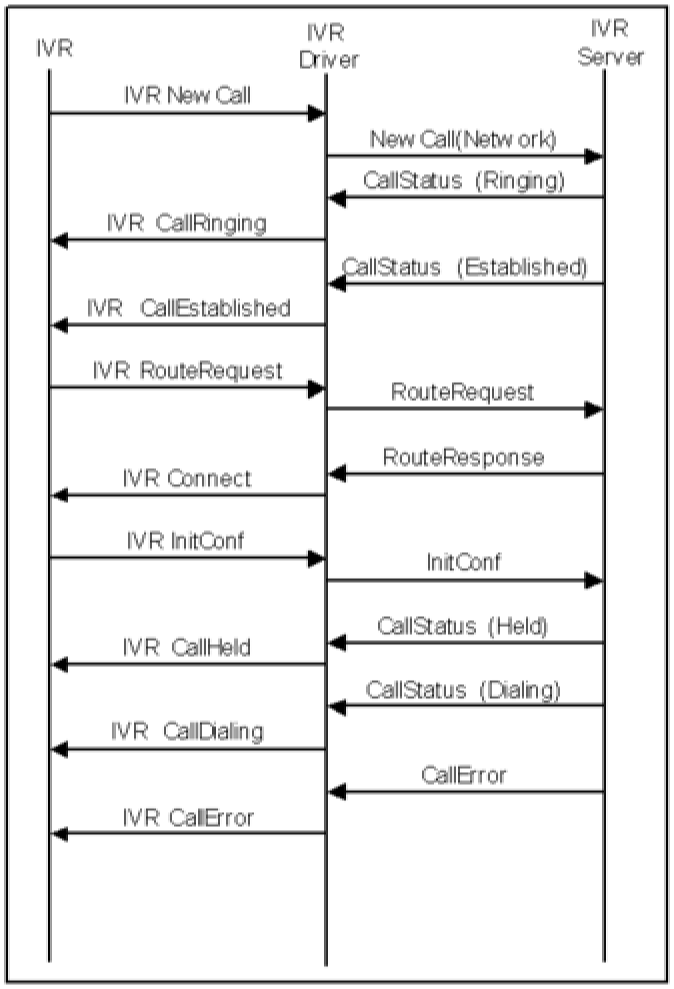
Conference Consult Call (Busy)



Conference Consult Call, Line Busy

A Busy response is not considered an error. When the party which is to be conferenced with the original caller is busy, the IVR driver must send a `RetrieveCall` message to retrieve the original call. Compare this to Conference Consult Call (Failed).

Conference Consult Call (Failed)



Conference Consult Call Failed Call Flow

If a `CallError` occurs, IVR Server automatically returns you to the same status as before the conference call was started. This means that the second call is terminated and the original call is retrieved without any input from the IVR.

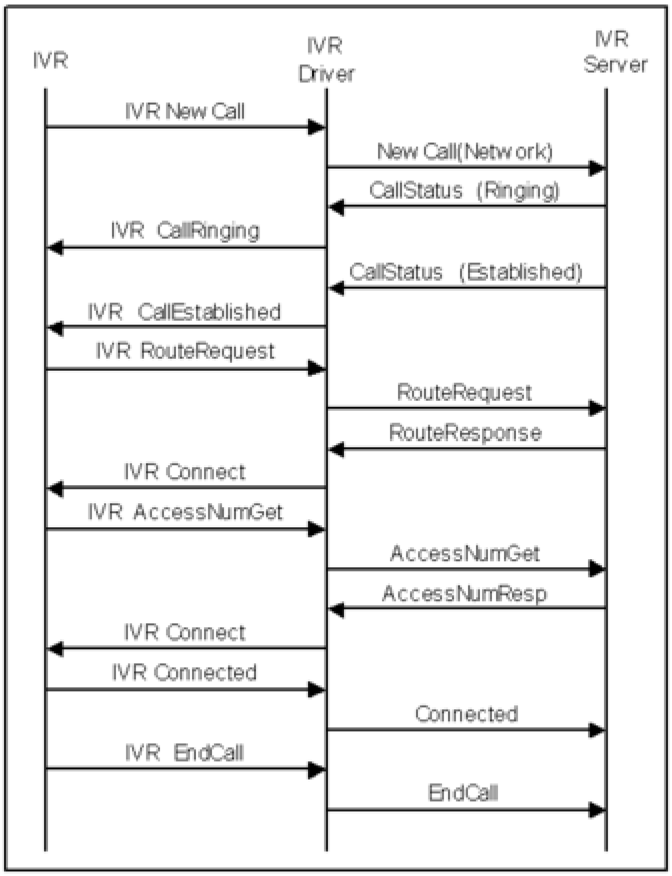
Important

If the IVR tries to retrieve the original call after a `CallError` message, the IVR will receive another error message because the original call has already been taken off hold.

Transfer Call Flow Diagrams

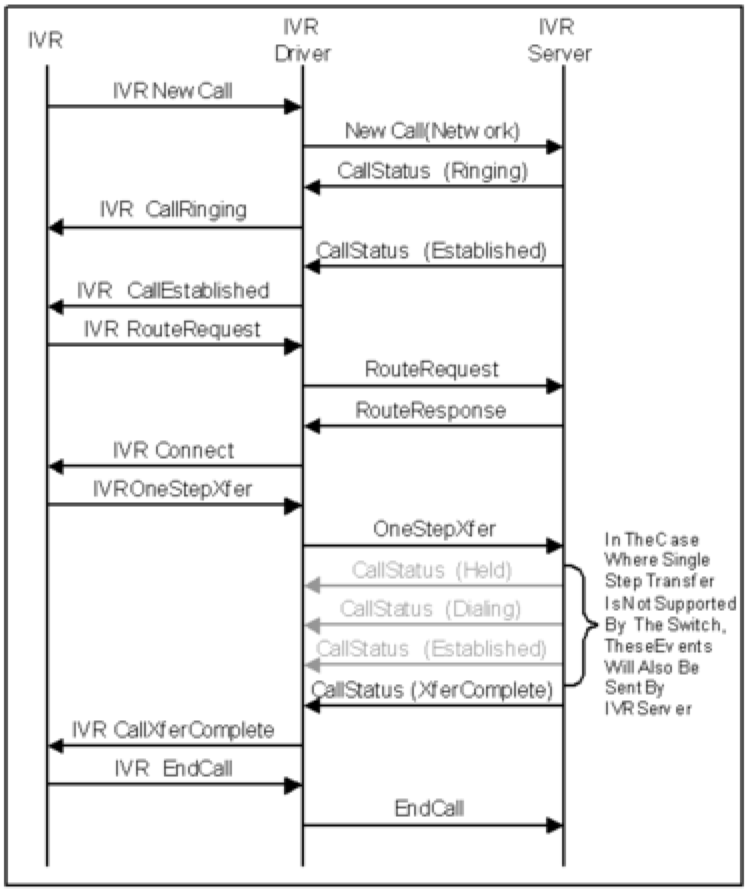
The following call flow diagrams illustrate several scenarios involving transferring calls.

Transfer to Remote Site



Transfer to a Remote Site

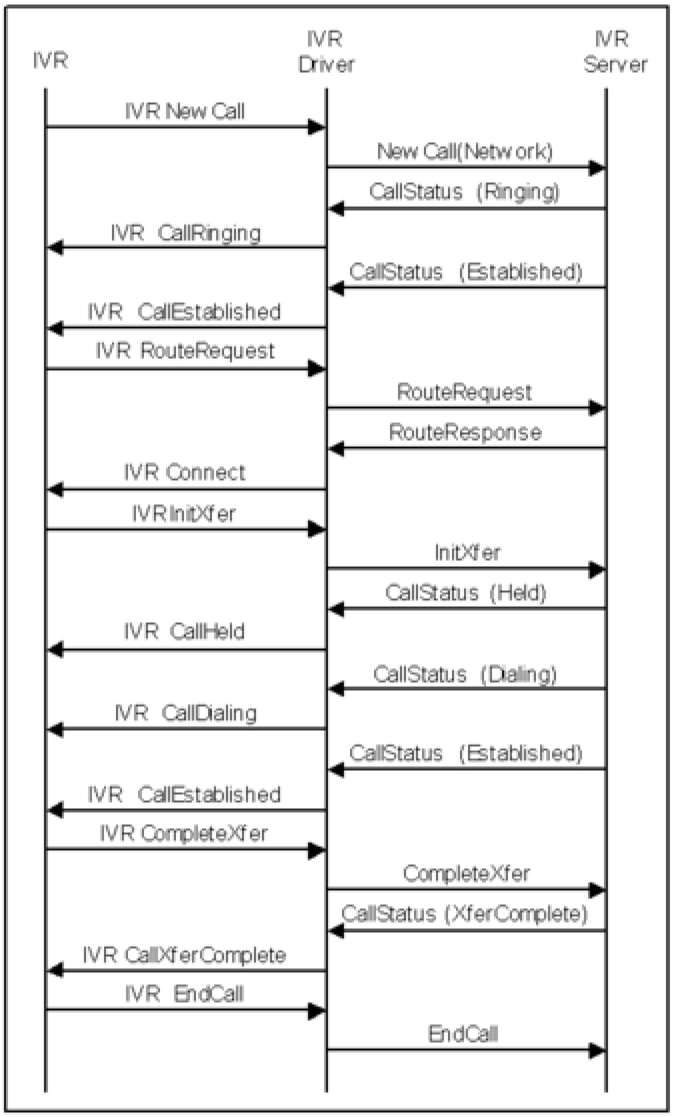
Single-Step Transfer



Single-Step Transfer

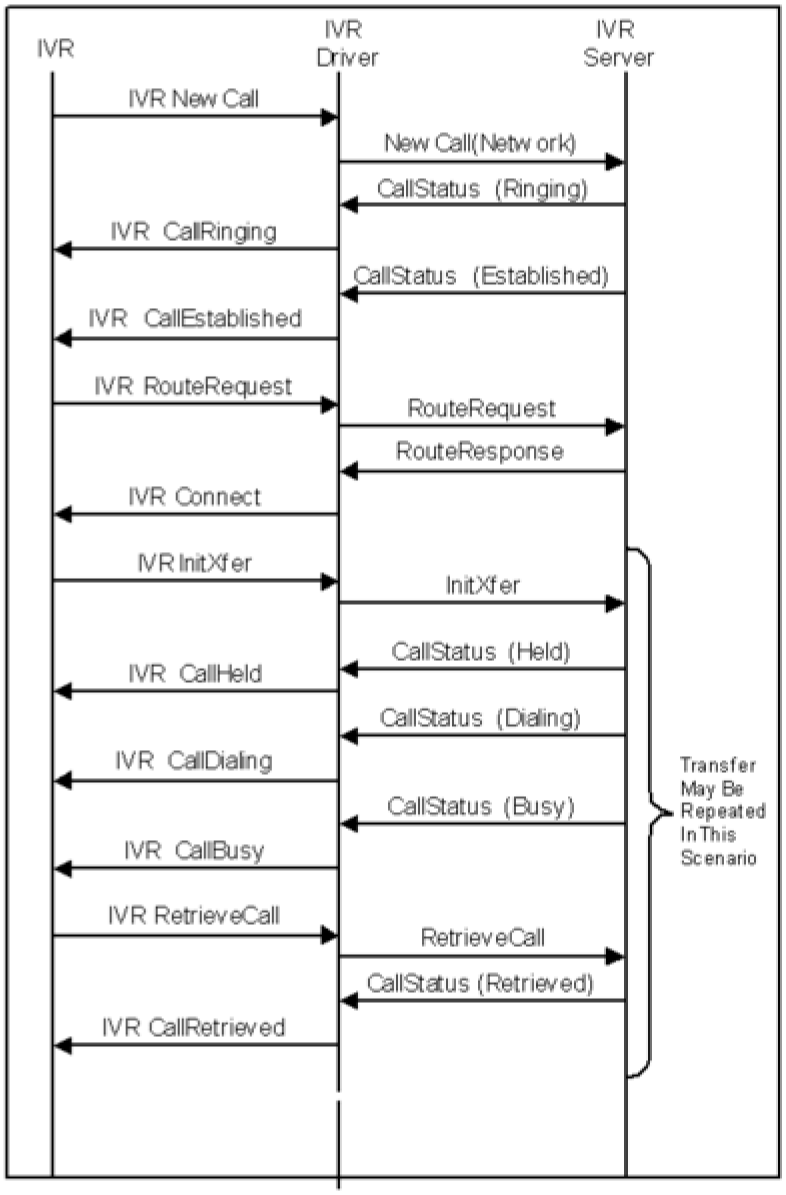
If a `CallError` occurs, IVR Server automatically returns you to the same status as before the transfer was started. This means that the original call is retrieved without any input from the IVR.

Transfer Consult Call



Call Flow for a Transfer Consult Call

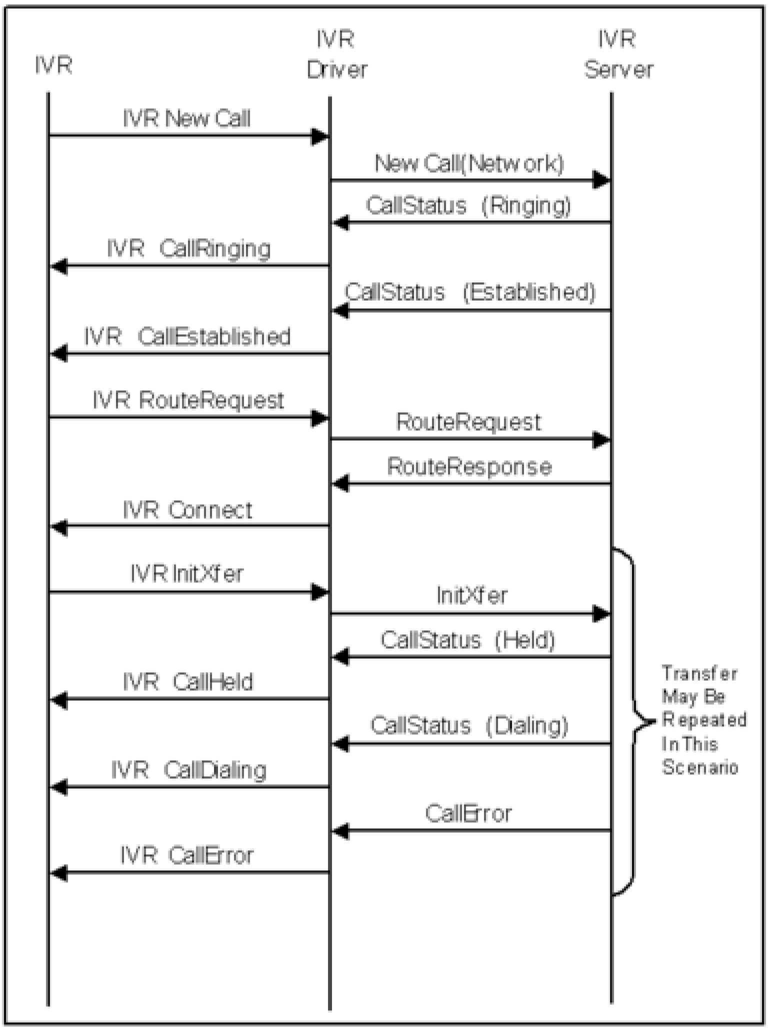
Transfer Consult Call (Busy)



Transfer Consult Call, Line Busy

A `Busy` response is not considered an error. When the party to which the original caller is to be transferred is busy, the IVR driver must send a `RetrieveCall` message to retrieve the original call. Compare this to Transfer Consult Call (Failed).

Transfer Consult Call (Failed)



Transfer of Consult Call Failed

If a `CallError` occurs, IVR Server automatically returns you to the same status as before the call transfer was started. This means that the second call is terminated and the original call is retrieved without any input from the IVR.

Important

If the IVR tries to retrieve the original call after a `CallError` message, the IVR will receive another error message because the original call has already been taken off hold.