IVR SDK 8.5 XML

Developer's Guide

About Genesys

Genesys is the world's leading provider of customer service and contact center software—with more than 4,000 customers in 80 countries. Drawing on its more than 20 years of customer service innovation and experience, Genesys is uniquely positioned to help companies bring their people, insights and customer channels together to effectively drive today's customer conversation. Genesys software directs more than 100 million interactions every day, maximizing the value of customer engagement and differentiating the experience by driving personalization and multi-channel customer service—and extending customer service across the enterprise to optimize processes and the performance of customer-facing employees. Go to www.genesys.com for more information.

Each product has its own documentation for online viewing at the Genesys Documentation website or on the Documentation Library DVD, which is available from Genesys upon request. For more information, contact your sales representative.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Genesys Telecommunications Laboratories, Inc., cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Genesys products.

Trademarks

Genesys and the Genesys logo are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other company names and logos may be trademarks or registered trademarks of their respective holders.

The Crystal monospace font is used by permission of Software Renovation Corporation, www.SoftwareRenovation.com.

Technical Support from VARs

If you have purchased support from a value-added reseller (VAR), please contact the VAR for technical support.

Customer Care from Genesys

If you have purchased support directly from Genesys, please contact Genesys Customer Care. Before contacting Customer Care, please refer to the ***Genesys Care Program Guide*** for complete contact information and procedures.

Ordering and Licensing Information

Complete information on ordering and licensing Genesys products can be found in the ***Genesys Licensing Guide.***

Released by

Genesys Telecommunications Laboratories, Inc. www.genesys.com

Document Version: 85sdk_dev_ivr-xml_06-2014_v8.5.001.00

Table of Contents

Preface

Welcome to the *IVR SDK 8.5 XML Developer's Guide*. This guide introduces you to the concepts, terminology, and procedures relevant to the Genesys IVR SDK XML, the tool for building drivers that allow your IVR (Interactive Voice Response Unit) to communicate with the Genesys IVR Server.

This document is valid only for the 8.5 release(s) of this product.

Note: For versions of this document created for other releases of this product, visit the Genesys Documentation website, or request the Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesys.com.

This preface contains the following sections:

For information about related resources and about the conventions that are used in this document, see the supplementary material starting on page 181.

About IVR SDK XML

In brief, this guide includes the following information:

- An overview of how IVR Server implements the Extensible Markup Language (XML) and of IVR architecture as it pertains to an XML-based client application.
- Sample call flows indicating request-response interactions for a variety of transaction types.

- Diagrams displaying client states and the transitions from each state to all other possible states.
- Detailed explanations of all Genesys IVR-specific XML messages and parameters.
- A step-by-step analysis of several common interaction types showing call flows and sample XML messages used to communicate between your IVR driver application and the Genesys IVR Server.
- The complete text of the `IServer.dtd` file.

New In Release 8.5

The following changes have been implemented in release 8.5:

- Support for new operating systems. Refer to the *Genesys Supported Operating Environment Reference Guide* for details.

Intended Audience

This guide, primarily intended for contact center administrators, contact center managers, operations personnel, and IVR developers, assumes that you have a basic understanding of:

- Computer-telephony integration (CTI) concepts, processes, terminology, and applications.
- Network design and operation.
- Your own network configurations.

You should also be familiar with the Extensible Markup Language (XML).

Usage Guidelines

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

- Creation of contact-center 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:
    a. The integration must use only published interfaces to access Genesys data.
    b. The integration shall not modify data in Genesys database tables directly using SQL.
    c. 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.

Making Comments on This Document

If you especially like or dislike anything about this document, feel free to e-mail your comments to Techpubs.webadmin@genesys.com

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this document. Please limit your comments to the scope of this document only and to the way in which the information is presented. Contact your Genesys Account Representative or Genesys Customer Care if you have suggestions about the product itself.

When you send us comments, you grant Genesys a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

Contacting Genesys Customer Care

If you have purchased support directly from Genesys, please contact Genesys Customer Care.

Before contacting Customer Care, please refer to the *Genesys Care Program Guide* for complete contact information and procedures.

Document Change History

This is the first release of the *IVR SDK 8.5 XML Developer's Guide.* In the future, this section will list topics that are new or that have changed significantly since the first release of this document.

Genesys

Part

1 Use of XML and Modes of Operation

Part One of the *IVR SDK 8.5 XML Developer's Guide* provides a general overview of the IVR SDK components that use XML, introduces the XML skills necessary in order to develop applications that communicate with those IVR components, and presents system requirements. It also explains the three modes of operation that IVRs use, how a mode is determined, and what operations are supported by each mode.

The information in Part One is divided between the following chapters:

- Chapter 1, "How IVRs Use XML," on page 17, explains how IVR components use XML, introduces basic XML skills, and presents system requirements.
- Chapter 2, "Modes of Operation for IVR," on page 23, explains the three modes of operation, how a mode is determined, and what operations are supported by each mode.

Chapter

1 How IVRs Use XML

This chapter provides a general overview of the IVR SDK components that use XML, introduces the XML skills necessary in order to develop applications that communicate with those IVR components, and presents system requirements. This chapter contains these sections:

How IVR Uses XML

The Genesys Interactive Voice Response (IVR) application programming interface (API) enables communication between a third-party IVR and the Genesys IVR Server. To establish this communication, you must create an application that functions as an IVR driver. This driver communicates with IVR Server using Extensible Markup Language (XML). IVR Server uses a customized *document type declaration* (DTD), that defines the Genesys-defined XML elements and attributes necessary to create call flows that are appropriate to your enterprise.

For the text of the DTD, see Appendix, “The IVR Server DTD,” on page 173.

XML Concepts

This guide assumes that you have a thorough understanding of XML data modeling. You should be familiar with the standards set in the XML specification v1.0. In addition, you need to know how to use an external (DTD) (in this case the `IServer.dtd` file located on your software CD).

IVR Architecture

Figure 1 shows how the XML interface connects the customer-developed IVR driver to IVR Server, which then communicates with Genesys Framework components.

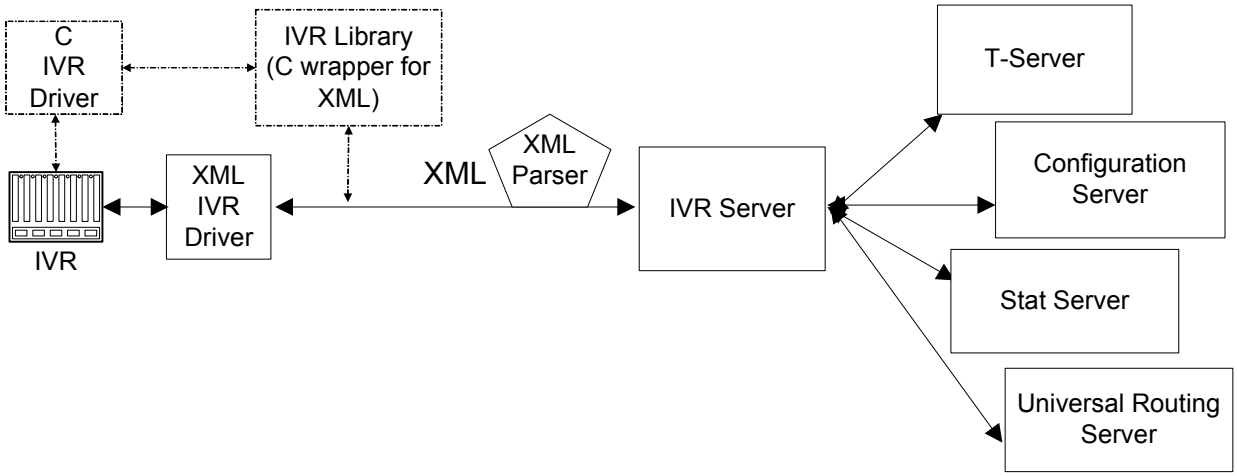

Figure 1: Genesys IVR Architecture

Stack Layers

Figure 2 contains a basic representation of the stack layers, that handle the communication among the IVR components: Descriptions of these stack layers follow:

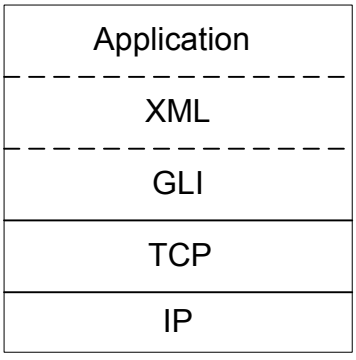

Figure 2: Stack Layers

- Application layer—Represents the XML-based client application built by the user to function as an interpreter between the IVR and the Genesys IVR Server.
- XML layer—Shows that XML is the language the application uses to communicate.

- GDI Link Interface (GLI) layer—Responsible for link-layer functions such as load balancing over multiple network interfaces and connection-failure detection using `Keep-Alive` messages. It is a proprietary transport protocol used to structure TCP/IP messages, and is a subset of the Generic Data Interface (GDI) protocol from Telcordia (formerly Bellcore). For details on the GDI, see "The GDI Specification".
- Transmission Control Protocol (TCP) layer—Provides transport functions.
- Internet Protocol (IP) layer—Provides routing capability.

The GDI Specification

The GDI protocol is a service-independent interface specification created by Telcordia Technologies, Inc. (formerly Bellcore). This protocol defines both link layer features (load balancing, link monitoring, and so on) as well as application/presentation layer features (ASN.1 data types and TCAP for session management).

The GLI Specification

The GLI protocol used by IVR Server is a strict subset of the GDI protocol, and contains only the link layer features of GDI described in Chapter 2 of the GDI specification. The application/presentation layer aspects of GDI related to TCAP and ASN.1 are not used by GLI and may be ignored. For GLI code examples, See "GLI Header" on page 95.

Reliability

The following notes indicate how the GLI subset of the GDI handles connections and supports keep-alive functions.

Link Failure The client initiates the TCP/IP connection to the Genesys IVR Server. In the event of a link failure, the client reinitiates the connection with the server. For details, see the GDI specification, Section 2.1.

Keep-Alive When the client receives a `Keep-Alive` request, it replies with `Keep-Alive` response. For the format of these messages, see the GDI specification, Sections 2.2, 2.2.2, and 2.2.3.

Error Messages When it receives an `Error` message, the client closes the current connection and initiates a new one. `Error` messages consist of the GDI header and an Error Code. For the definition of these Error Codes, see the GDI specification, Section 2.2.4.

Errors can be generated in the following scenarios:

- The client is not authorized to connect to the server (client access is enabled by using the Genesys Configuration Manager).
- The version number (part of the GDI header) is invalid.

Multiple Circuit Connections

The client can use Multiple Circuit Connections to connect with the Genesys IVR Server. For details, see the GDI specification, Section 2.3.

Security

Only the server side enforces security implementation. Security implementation is not included in the client. For details, see the GDI specification, Section 2.4. The server side may also enforce Transport Level Security (TLS) which is the industry standard protocol for secure communications on the Internet and the successor of SSL 3.0. For detailed information refer to the *Genesys 8.0 Security Deployment Guide*.

Timeout

The client generates a timeout in the event of a lack-of-response condition of the server. For details, see the GDI specification, Section 2.2.4.

XML Message Guidelines

The following points indicate important message considerations.

Note: Keys in Key/Data pairs cannot include colons ":". An error message appears if keys include this character.

Special Character Encoding

The characters <, >, &, ", and ' must be coded in escape form in order for the parser to interpret them correctly. This applies to all portions of the messages. Table 1 lists the escape forms:

Table 1: Escape Form for Certain Characters

Character	Escape Form
<	<
>	>
&	&
"	"
'	'

Note: The messaging protocol used between IVR Server and IVR Clients (for instance, Genesys IVR Drivers, Genesys Voice Platform [GVP], and custom-built SDK clients) is XML, and certain character codes are not valid in an XML document. Characters with a value less than hexadecimal number `0x20` are not valid, with the exception of the characters `0x09`, `0x0A`, and `0x0D`. These exceptions correspond to the ASCII control characters `TAB`, `LINE FEED`, and `CARRIAGE RETURN`. Application user-data keys or values should not contain any of the disallowed character codes.

Header Structure For the GDI header structure, see the GDI specification, Section 2.2.

Data Message Type The XML-encoded application layer messages are considered *data messages.* The value for this message type is `03`. Data messages are sent from the client to the Genesys IVR Server.

MessageLength The `MessageLength` includes the one-byte `VersionNumber` and the one-byte `ApplicationID` in its count. Key/Data pairs to be passed to T-Server are limited to 16 K in size. Other messages, including the header, can be 64 K in size. For details, see the GDI specification, Section 2.2.

The CallId Parameter The IVR driver assigns a `CallId` value at the time of the initial `NewCall` or `MakeCall` request. This `CallId` may be present in subsequent message for this call. The value can be any valid character string.

Note: The `CallId` parameter must be unique for every concurrently active call handled by an IVR Server application. This means not only that each individual client application has to generate unique `CallIds`, but if multiple clients connect to the same IVR Server application, those clients must ensure that they do not use the same `CallIds`.

Error Message Type The client can receive `Error` messages from the server. The message type for the `Error` message has a value of `02`.

Keep-Alive Message Types The message type of a `Keep-Alive` request has a value of `00`. The message type for the `Keep-Alive` response has a value of `01`.

VersionNumber The `VersionNumber` must have a value of `2`.

ApplicationID Value The only valid value for the `ApplicationID` is `0`.

Attached Data

Message extensions are optional parameters that are included in some messages. They can be used by routing strategies, IVR scripts, and the client application. The Genesys IVR Server does not interpret these values, it simply forwards them between the Genesys Universal Routing Service (URS) and the client application.

Attaching CED

CED is an optional parameter for `RouteRequest` and `TreatStatus` messages. If provided, the value of CED will be present in the `AttributeCollectedDigits` parameter of the T-Library message that the Genesys IVR Server forwards back from the Genesys T-Server. The TLib message is either `EventRouteRequest` or `EventTreatmentEnd`, depending on which message started the sequence.

The UDataEx Parameter

`UDataEx` is an optional parameter that you can include in almost every message from your client application to the Genesys IVR Server. When IVR Server receives data in the `UDataEx` parameter, it attaches the data to the call as user data, thus providing a convenient way to combine attached data with another request. For example, sending `RouteRequest` with `UDataEx` attached is equivalent to calling `UDataSet`, followed by a `RouteRequest`.

Sockets, Ports, Channels, and DNs

For both IVR In-Front and IVR Behind configurations, IVR Server uses a list of DNs and IVR ports that correspond to a particular IVR. The association between IVR ports and DNs is defined in the Configuration Layer, and it is used by IVR Server to correlate a given IVR port number (supplied by a client) with a DN configured on the switch. (See the *IVR Interface Option 8.0 IVR Server System Administrator's Guide* for more information.) The IVR port related to a call is specified as the `calledNum` parameter in the `NewCall` XML message. (See “NewCall” on page 99.)

All of the message exchanges for a given IVR and IVR Server pair can be made using a single TCP/IP socket connection, and this is the recommended approach. It is possible to implement alternative connection handling methods, but special care must be taken when doing so.

Genesys

Chapter

2 Modules of Operation for IVR

This chapter provides a general overview of the IVR SDK's three modes of operation (Behind, In-Front, and Network), how a mode is determined, and what operations are supported by each mode.

This chapter contains these sections:

IVR Server High-Availability Modes

Table 2 summarizes high-availability options supported by IVR Server.

Table 2: IVR Server High-Availability Modes

Type	Definition
Load Balancing	IVR driver assigns (and tracks) XML calls to more than one IVR Server
Hot Standby	IVR driver writes all XML messages to both the primary and backup IVR Servers
Warm Standby	IVR driver writes and reads to the current primary IVR Server
No HA	IVR driver writes and reads all XML for all calls to and from one IVR Server

Load Balancing

The IVR driver sends the `Login` message to each of *N* servers and then decides how to map an IVR call to a single IVR Server. Typically, round-robin is chosen. If an IVR Server fails and drops its connection, the mapping is changed to send subsequent calls to the available IVR Server(s). The failed IVR Server is added to a reconnect list, from which it is returned to the pool of available servers. After reconnecting, the `Login` must be repeated.

Once a call is started on a chosen IVR Server, it must remain there. All calls on the failed IVR Server are lost.

Hot Standby

The IVR driver must send all XML requests to both the primary and backup IVR Servers, with exactly the same information. The current primary IVR Server responds to the request. The only exception to this is the `LoginReq`, `LoginResp`, and `FlowControl` messages for the In-Front and Behind operation modes, in which case both the primary and backup IVR Servers will respond. If the connection to the primary IVR Server fails, the driver continues to send messages for all calls to the backup IVR Server which, after failure of the primary server, will start sending the return messages.

Warm Standby

The IVR driver sends the `Login` message and subsequently all calls from the IVR to the current primary IVR Server. When that connection fails, the IVR driver attempts to connect to the backup IVR Server. (The `Login` will be necessary.) All calls on the failed IVR Server are lost.

No HA

The IVR driver sends the `Login` message and subsequently all calls from the IVR to one IVR Server.

Note: See the *IVR Server System Administrator's Guide* for details on how the IVR Server is configured and works in these modes, and advantages, disadvantages, and limitations. Note that Load Balancing and Hot Standby modes are mutually exclusive.

IVR Server Operational Modes

A running IVR Server application has three different modes in which it may operate:

- The IVR *Behind*-the-switch mode is a basic configuration in which a T-Server connected to the premise switch can monitor the call activity on IVR channels.
- In IVR *In-Front*-of-the-switch mode, a Computer Telephony Integration (CTI) link is not involved with the call processing.
- In IVR *Network* mode, the IVR Server (an IVR T-Server running in Network mode) is a link to a user-provided Network IVR application. A routing strategy and a Genesys Network T-Server route calls to the Network IVR for processing.

In this document, these three modes are referred to as In-Front, Behind, and Network, respectively.

For more information regarding these modes of operation and their configuration, refer to *The IVR Interface Option 8.5 IVR Server System Administrator's Guide.*

Depending on the mode in which your IVR Server is deployed, call flows vary, and certain messages may, or may not be supported.

Implications of the Different Modes

Generally speaking, In-Front and Behind modes behave identically. The exception is that with In-Front, IVR Server is a client of itself. By contrast, with Behind mode, the IVR Server is a client of another T-Server. This has repercussions in that some premise T-Servers support functionality that IVR Server's T-Server does not, such as conferencing. On the other hand, Network mode is at times dramatically different and more akin to the GenSpec XML-based Network T-Server. (See any Network T-Server Deployment Guide for details.)

Modes and Their Uses of Interfaces

IVR Server contains several different client and server interfaces for use during operation. Although technically all of the interfaces are available, regardless of mode, logically they are not. The relevant interfaces are:

- XML/GLI server—Provides access to XML-based clients, such as IVR drivers.
- T-Server—A fully functioning Genesys T-Server that generates various T-Library events.
- T-Library client—A T-Server client for interfacing with other Genesys T-Servers.

- `StatServer` client—Interfaces with Genesys Stat Server to provide statistics lookup.
- `ConfigServer` client—Client for establishing and maintaining configuration information relevant to IVR Server.

Behind Mode and Interfaces

The T-Server interface is not utilized and should be considered unavailable if your have deployed Behind mode. For this particular mode, IVR Server is a client of a foreign T-Server—generally, a premise T-Server. It is important to note that any T-Server events, such as `EventRouteRequest`, cannot be sent in a Behind mode call since this is a function of the T-Server interface.

In-Front Mode and Interfaces

When the mode for a given call is In-Front, all interfaces are active (or available, depending on the specific configuration). In this case, the T-Library client interface connects to the T-Server interface, and IVR Server becomes a client of itself. This can cause some confusion in reading logs, because messages seem to appear more often than they should.

Network Mode and Interfaces

The T-Library client interface is not used and requires no configuration for Network mode. Additionally, if IVR Server is running only in Network mode, an IVR Server application is not required in Configuration Manager. If this setup is chosen, the `StatServer` and `ConfigServer` interfaces will also be offline.

Determination of Mode

At any given time, an IVR Server application is in one of the three types of modes. This means that individual calls are associated with a particular mode at call origination (either `NewCall` or `MakeCall`). You can determine the mode by using the `CalledNum` value of `NewCall` or the `OrigNum` value of `MakeCall` messages. Figure 3 provides a diagram of the decision process.

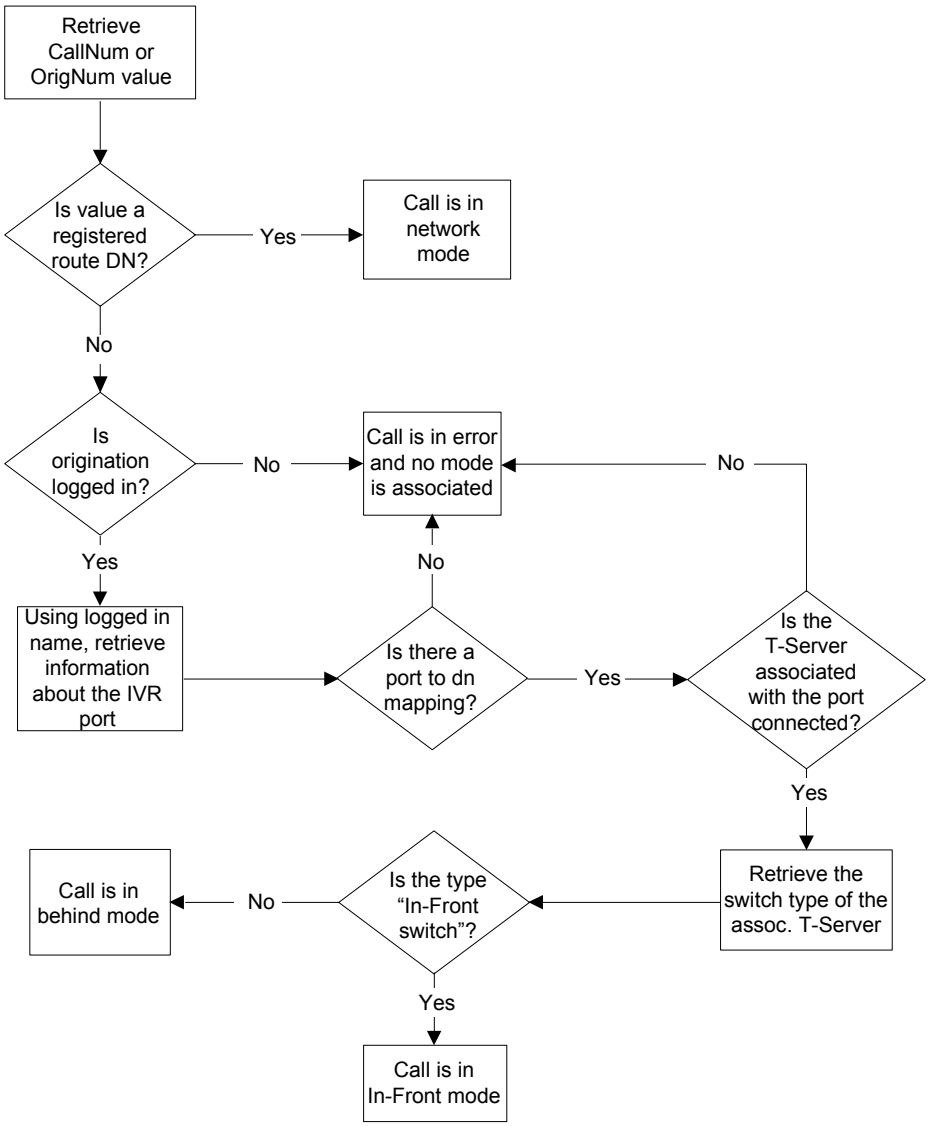

Figure 3: Mode Determination Process

When a call is in error, an `EndCall` message with `EndCause=FeatureNotSupported` is returned. This particular type of message usually indicates a configuration error. The decision block that determines whether a T-Server is connected relates to the `Connections` tab of the IVR Server `Application` object, not a network connection.

Individual Message Support and Behavior

Supported Messages

After you determine the mode of the call, any subsequent messages received for that call behave according to the call model for its type. Table 3 lists all the IVR Server XML client messages, and indicates whether they are supported for each type.

Table 3: XML Message Support

XML Message	PGF Event Name	Supported		
		In-Front	Behind	Network
AccessNumCancel	Cancel Call Data Transfer	Yes	Yes	No
AccessNumGet	Call Data Transfer	Yes	Yes	No
CallInfoReq	Call Info Request	Yes	Yes	Yes
CancelCompleted	Cancel	Yes	Yes	Yes
CompleteXfer	Consult Conference	No	Yes	No
Connected	Connected	Yes	Yes	Yes
EndCall	End Call	Yes	Yes	Yes
Failure	Route Failed	Yes	Yes	Yes
GetStatReq	Get Stat	Yes	Yes	Yes
InitXfer	Consult Transfer	No	Yes	No
LoginReq	Login Request	Yes	Yes	Yes
LogMsg	Log Message	Yes	Yes	Yes
MakeCall	Make Call	No	Yes	No
NewCall(CallControlMode=Genesys)	New Call Genesys	No	No	Yes
NewCall(CallControlMode=Network)	New Call Network	Yes	Yes	Yes
OneStepConf	Single Step Conference	No	Yes	No
OneStepXfer	Single Step Transfer	No	Yes	No
PeekStatReq	Peek Stat	Yes	Yes	Yes

Table 3: XML Message Support (Continued)

XML Message	PGF Event Name	Supported		
		In-Front	Behind	Network
Reset	N/A	N/A	N/A	N/A
RetrieveCall	Retrieve	No	Yes	No
RouteRequest	Route Request	Yes	Yes	Yes
TreatStatus (Status=Completed)	Treatment Complete	Yes	Yes	Yes
TreatStatus (Status=NotStarted)	Treatment Not Started	Yes	Yes	Yes
TreatStatus (Status=Started)	Treatment Started	Yes	Yes	Yes
UDataDel	Delete User Data	Yes	Yes	No
UDataGet	Get User Data	Yes	Yes	Yes
UDataGetAll	Get All User Data	Yes	Yes	Yes
UDataSet	Update User Data	Yes	Yes	Yes

Mode Behavioral Differences

Support of a message in a particular mode does not mean that this message will prompt the same behavior by the server in all modes. In addition, some supported messages might be required for proper operation of one mode, but might be entirely optional for another. This section details those messages that exhibit exceptions across modes.

NewCall (CallControlMode=Network)

In Network mode, no `CallStatus` messages are generated because there are no agents or extensions involved. With In-Front mode, `ringing` and `established` status messages are always generated. However, in Behind mode, they are dependent on `EventRinging` and `EventEstablished` events occurring on the associated premise T-Server. If the premise T-Server generates both `EventRinging` and `EventEstablished` for calls directed to the IVR, the behavior for In-Front and Behind modes is the same.

RouteRequest

Network mode uses route type DNs for all call processing. Routing requests are generated on the same number as `CalledNum` from the `NewCall` message. In addition, Network mode calls can use a subset of the called number as the

actual DN. (See the `called-num-subset` option in the *Administrator's Guide*.) This setting is not used in In-Front or Behind mode. Also, the `RouteDN` attribute of `RouteRequest` is required for In-Front and Behind modes, but it is disregarded in Network mode.

In Network mode, only a single `RouteRequest` can be made for a call. However, since it is possible for routing failures to be indicated, target selection can occur multiple times for the same `RouteRequest`.

Since routing failure conditions cannot be indicated for In-Front or Behind mode (see "Failure"), these modes support multiple `RouteRequest` / `RouteResponse` interactions for a call.

Connected

This message must be sent for a Network mode call in order to properly complete the routing interaction. For In-Front and Behind modes routing cannot fail, and the use of the `Connected` message is highly discouraged. Sending the `Connected` message for In-Front or Behind mode calls will cause the routing interaction to be canceled. This is used to implement the `Route Abort` feature that is required for compatibility purposes in some IVR drivers, although it is considered deprecated.

Failure

The `Failure` message maps to `EventError` in Network mode, and it can be used to indicate routing problems. For In-Front and Behind modes, routing cannot fail, and the use of the `Failure` message is not recommended.

LoginReq

This message is necessary for proper determination of In-Front and Behind modes.

Genesys

Part

2 IVR Server In-Front and Behind Mode

Part Two of the *IVR SDK 8.5 XML Developer's Guide* familiarizes you with the Genesys IVR SDK XML, the tool for building drivers that allow your IVR (Interactive Voice Response Unit) to communicate with the Genesys IVR Server. This portion of the guide introduces you to the relevant concepts, terminology, and procedures used by the Genesys IVR SDK XML in standard integration situations.

The information in Part Two is divided among the following chapters:

- Chapter 3, "IVR In-Front and Behind State Machine Diagrams," on page 33, contains state machine diagrams from the viewpoint of the IVR driver. This chapter also includes messages that can be sent from any state, error messages, and the procedure for configuring the statistics used in the statistics messages.
- Chapter 4, "In-Front and Behind Call Flow Diagrams," on page 43, provides call flow diagrams for many common scenarios. It is intended as a reference.
- Chapter 5, "IVR XML Protocol Messages and Parameters," on page 61, contains tables showing the parameters for each message, the message direction, and whether the parameters are required or optional.
- Chapter 6, "Using the IVR XML Protocol: Examples," on page 95, presents sample XML messages and comments for a number of commonly-encountered call flow scenarios. These sample XML messages are intended as starting points for building your IVR driver application.

Genesys

Chapter

3 IVR In-Front and Behind State Machine Diagrams

This chapter includes diagrams showing sequences of events and transitions from state to state in a standard application using an IVR driver. This chapter contains these sections:

Overview

The following diagrams outline the states available to the client-server system, shown from the perspective of the client IVR driver application.

Events in the diagrams that are prefaced with “IVR” are generated by the IVR and sent to the client IVR driver application. All other events are sent from the IVR Server to the IVR driver.

Note: The messages in these diagrams are designed to represent typical messages that your IVR sends to your IVR driver client application. The messages might differ somewhat from those given below, depending on the IVR hardware and software your enterprise uses. Notice that the `EndCall` message can arrive at anytime from the IVR after the call has started.

The first diagram in this chapter, Figure 4 on page 35, shows the initiation of a call and marks the points where a call treatment, transfer, or conference could begin. Remote transfers, local transfers, conferencing, and call treatments are shown in separate state machine diagrams that follow this Call Routing example.

The final state machine diagram, Figure 8 on page 39, shows a `MakeCall` call flow. With `MakeCall`, the IVR system (or an outbound calling program that uses the IVR) initiates the call. This is an alternative to `NewCall` and is therefore shown separately.

Call Routing States

Figure 4 on page 35 includes the initial `NewCall` messages that lead to the `Call Started` or `Connecting` state—the starting points for transfers and conferencing—followed by the IVR driver states found in a routing call flow.

`NewCall` is the starting point for the subsequent diagrams and tables, with the exception of the `MakeCall` call flow (“Make Call States” on page 39).

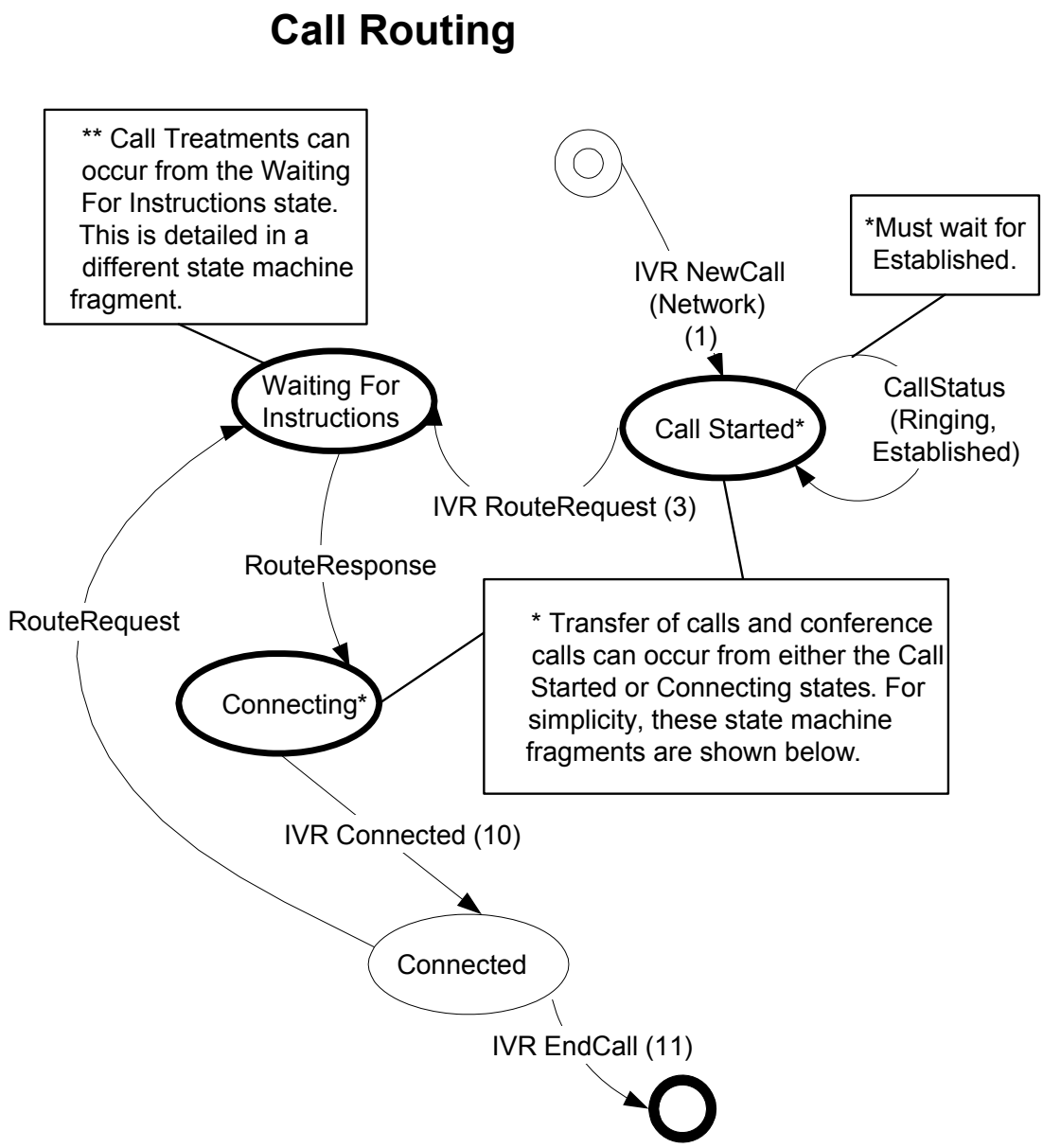

Figure 4: Call Routing States Diagram

Transfer States

The following diagram shows the IVR driver states encountered during a transfer. Throughout these transfer sequences, the IVR and the Genesys IVR Server communicate through your IVR driver application.

Note: The transfer events and their branches, shown in Figure 5 on page 36, are only available when the IVR Server is running in Behind mode.

The IVR Server supports two types of transfers, one-step and consult. One-step transfers are made as quickly as the switch can perform them. They are most useful for predictive dialing situations. Consult transfers place the original call on hold and establish that the transfer line is available before completing the transfer.

Note: Not all switches support one-step transferring. If you receive an "Unsupported Operation" error message when making a one-step transfer, use the consult transfer instead. In case of an error message, the original call is reactivated.

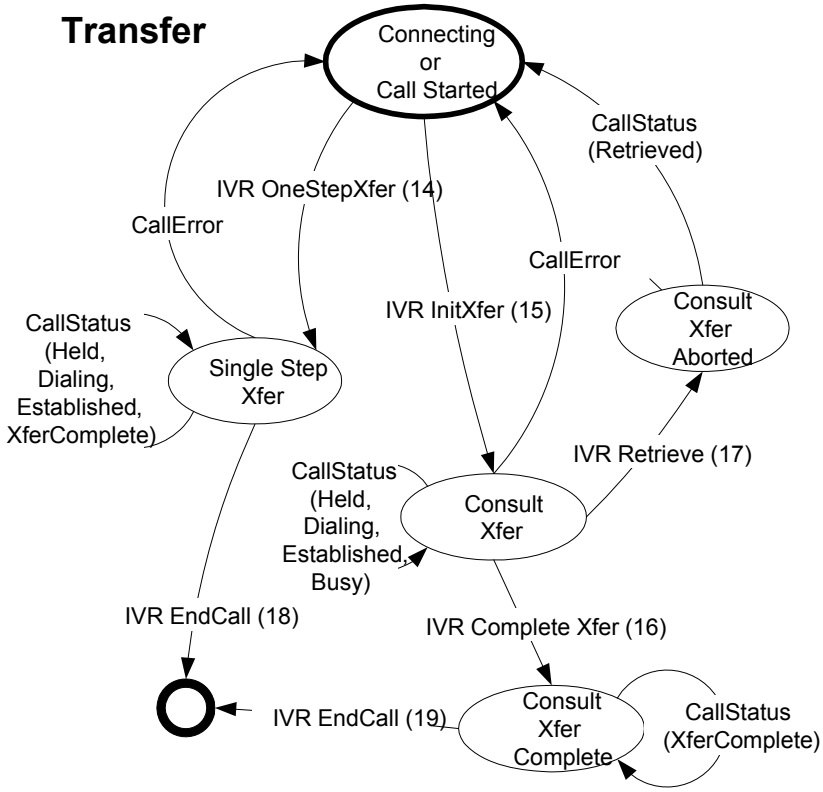

Figure 5: Transfer State Machine

Conference States

The following diagram shows the IVR driver states encountered when conferencing a call. Throughout these conferencing sequences, the IVR and the IVR Server communicate via your IVR driver application.

Note: The conference events and their branches, shown in Figure 6 on page 37, are only available when the IVR Server is running in IVR-Behind mode.

The IVR Server supports two types of conferencing, one-step and consult. One-step conferences are made as quickly as the switch can perform them. Consult conferencing places the original call on hold and establishes that the new conference line is available before completing the conference.

Note: Not all switches support one-step conferencing. If you receive an "Unsupported Operation" error message when attempting a one-step conference, use the consult conference method instead. In case of an error message, the original call is reactivated.

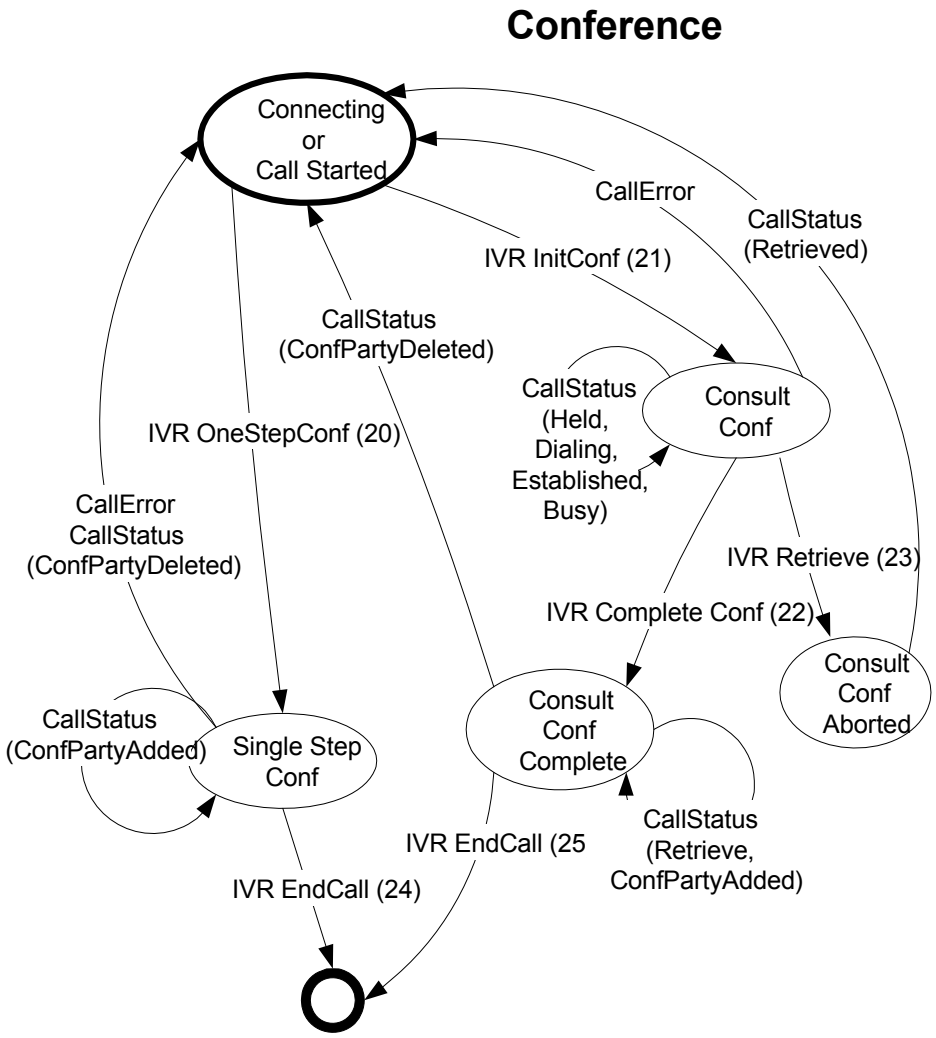

Figure 6: Call-Conferencing States

Call Treatment States

A call treatment is an operation performed on a call and can be one of a number of types. These types can be sequences of automated questions that collect caller information, music on hold, or some other handling of the call. Invoking a call treatment application involves the states shown in the following diagram, Figure 7.

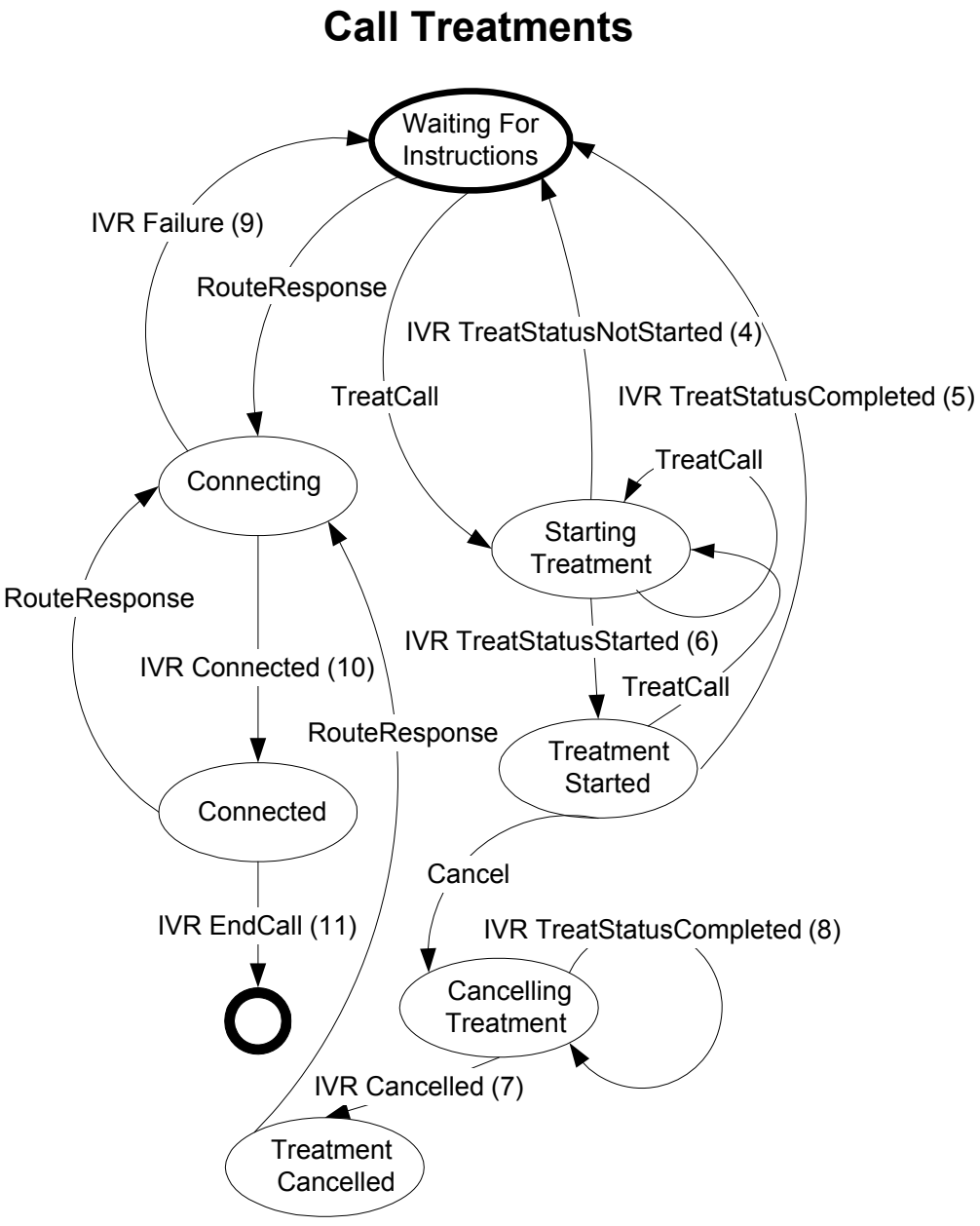

Figure 7: Call Treatment States

Make Call States

The `MakeCall` operation is an alternative to the `NewCall` scenario. In a `MakeCall` call flow, the initial call is outbound rather than inbound. Figure 8 shows the states that can occur during an `MakeCall` operation.

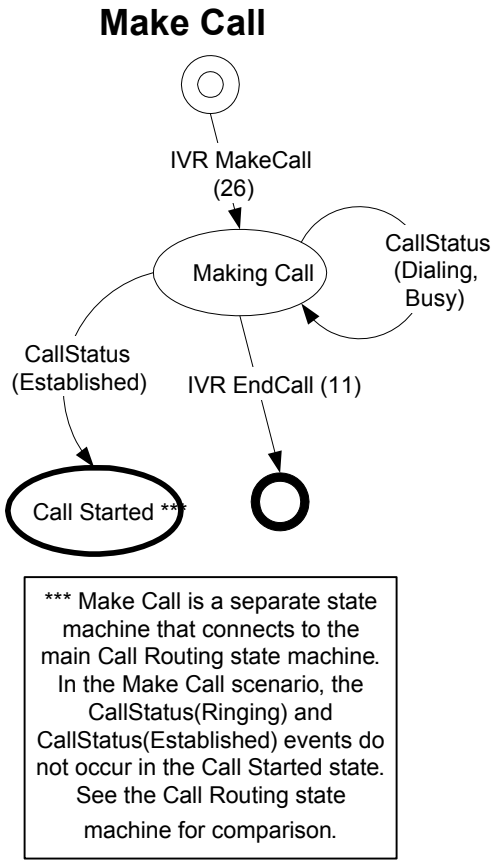

Figure 8: MakeCall States

Additional Event Messages

In addition to the events included in the diagrams above, you can have the IVR driver send the following messages to the IVR Server.

Anytime Messages

Messages that can be sent at any point in a call after the `NewCall` message:

`PeekStatReq`—Requests statistical information from Stat Server. You can request two statistics, `CurrNumberWaitingCalls` and `ExpectedWaitTime`. These allow you to inform the caller how long the expected wait time is.

`GetStatReq`—Requests statistical information from Stat Server. You can request two statistics, `CurrNumberWaitingCalls` and `ExpectedWaitTime`. These allow you to inform the caller how long the expected wait time is.

Note: These statistics must be configured in the Stat Server application. See "Configuring Stat Server Statistics" on page 42 for the statistic parameters.

`LogMsg`—Writes data to a log file. This file can be local or on the IVR Server. Set the log location in the `Data Options Transport` section of the IVR Server `Application` in Configuration Manager.

After CallStatus (Ringing) Messages

Messages that can be sent after the IVR Server has sent a `CallStatus (Ringing)` message:

`UDataSet, UDataGet, UDataGetAll, UDataResp, UDataDel`—These user data messages allow you to set up and manipulate key/value lists that are stored in T-Server for the duration of an interaction. These lists can include multiple key/value pairs that contains customer data such as current account information. The IVR requests this information, which the driver forwards to T-Server by way of the IVR Server.

- `UDataSet`—Requests T-Server to add specified key/value pairs to the call object. To replace data, use `UDataSet` with Action set to `Replace`.
- `UDataGet`—Requests data for an existing key or keys.
- `UDataGetAll`—Requests data for all existing keys.
- `UDataResp`—The response to a previous `UData` request.
  - If the request was Get and Result was set to `Success`, the data is in the response.
  - If the request was `Set` or `Del` and `Result` was set to `Success`, the operation was successful and no data is returned.
  - If the operation was not successful, it failed for the specified reason: `NoSuchCall, NoMatch` (no such key), `FeatureNotSupported` (cannot add key/value pairs), or `MiscError`.
- `UDataDel`—Removes one or all key/value pairs for a call. To remove one, use Action set to `DeleteKey`. To delete all keys, use Action set to `DeleteAll`.

`CallInfoReq`—A request from the IVR for T-Server call data, forwarded to T-Server by the IVR driver via the IVR Server. The data can include:

- `ANI`—The calling party's number.
- `DNIS`—The number or switch alias called by the caller.
- `CalledNum`—The IVR port number of the call. The port numbers are configured in the IVR application in the Configuration Layer.

- `ConnID`—The connection number attached to the IVR port for the call. Framework uses this number as a reference for all parties involved with the call.
- `TSCallId`—The call ID assigned to the call.
- `PortDN, PortTrunk, PortQueue`—The directory number, trunk and/or queue of the IVR port.
- `OtherDN, OtherTrunk, OtherQueue`—The directory number, trunk and/or queue of the party on the other end of the call from the IVR port.
- `LastEvent`—The last T-server event received for the IVR port, such as `EventRinging, EventEstablished,` and so on.
- `FirstHomeLocation`—This attribute corresponds to the T-Library's `AttritributeFristTransferHomeLocation` attribute.

Note: If you send one of these six messages before the `CallStatus (Ringing)` message, the request simply times out.

Error Messages

The IVR can return one of several types of error messages:

- `Failure`—If a connection attempt is unsuccessful, the IVR sends a `Failure` message with the appropriate reason attached: `Busy, NoAnswer,` or `ConnectFailed.`
- `CallError`—Returned when a request fails. See the DTD for a full list of reasons that might be attached to the message. The `CallError` message may also include a `TlibErrorCode`. For a list of T-Library error codes and their explanations, see a T-Server deployment guide.
- `Messages`—Requests can generate responses indicating that an operation failed if the request referred to invalid or non-existent objects or was incorrectly formatted. The response indicates the nature of the problem.

Note: If you encounter an error during conferencing or transferring, the IVR Server reactivates the original call (if it was on hold for a consult transfer or conference) or returns it to the original port (in the case of one-step transfers and conferences).

Configuring Stat Server Statistics

To configure the `ExpectedWaitTime` and `CurrNumberWaitingCalls` statistics:

1. Open the Stat Server application in Configuration Manager.
2. Click the `Create New Section/Option` icon on the `Options` tab menu.
3. Enter the statistic name and click `OK`.
4. Locate the new statistic in the list and double-click it.

   A blank pane appears.
5. Right-click and select `New` from the pop-up menu.
6. Enter the object names and values, one set at a time, as given in Table 4.
7. Click `OK` to exit application configuration and save the new settings.

For more information on configuring Stat Server, see the *Stat Server User's Guide*.

Table 4: Stat Server Statistic Values

Statistic Name	Objects	Category	MainMask	Subject
CurrNumberWaitingCalls	Queue, RoutePoint, GroupQueues	CurrentNumber	CallWait	DNAction
ExpectedWaitTime	Queue, RoutePoint, GroupQueues	ExpectedWaitTime	CallWait	DNAction

Genesys

Chapter

4 In-Front and Behind Call Flow Diagrams

This chapter provides detailed information about specific, selected paths through the call control state machine described in Chapter 3, “IVR In-Front and Behind State Machine Diagrams,” on page 33. These paths have specific relevance to In-Front and Behind mode deployment.

This chapter contains these sections:

Overview

The call flow diagrams available in this chapter are intended to be used as reference. Some of these call flows are expanded versions of the call flows analyzed in Chapter 6, “Using the IVR XML Protocol: Examples,” on page 95. The remaining call flow diagrams illustrate the request-response sequences for additional interaction types.

You can find complete lists of the conference and transfer call flow diagrams under “Conference Call Flow Diagrams” on page 49 and “Transfer Call Flow Diagrams” on page 55.

Call Routing Call Flow

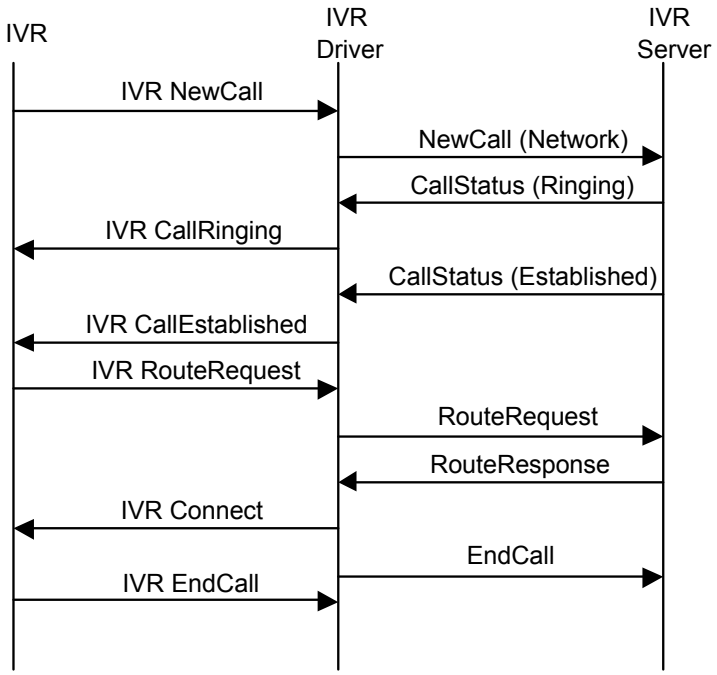

Figure 9: Call Routing Call Flow

See “Routing” on page 104 for sample XML messages and comments on the IVR driver/IVR Server segment of this call flow.

Call Treatment

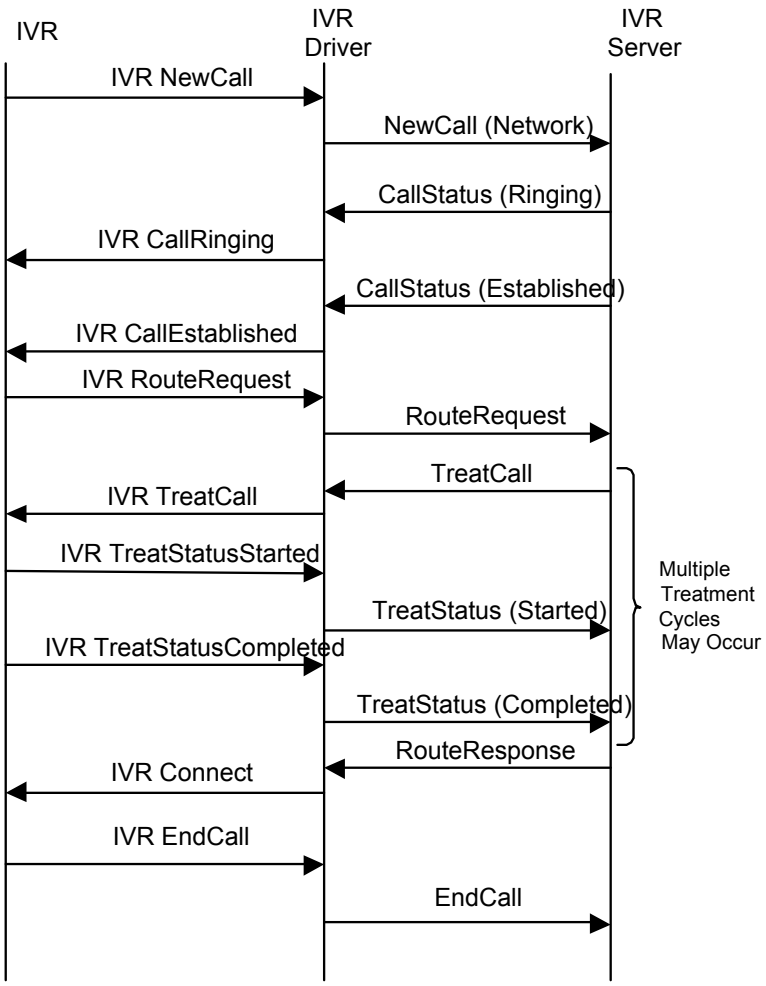

Figure 10: Call Treatment Call Flow

See “Call Treatment Operation” on page 107 for sample XML messages and comments on the IVR driver/IVR Server segment of this call flow.

Call Treatment Failed

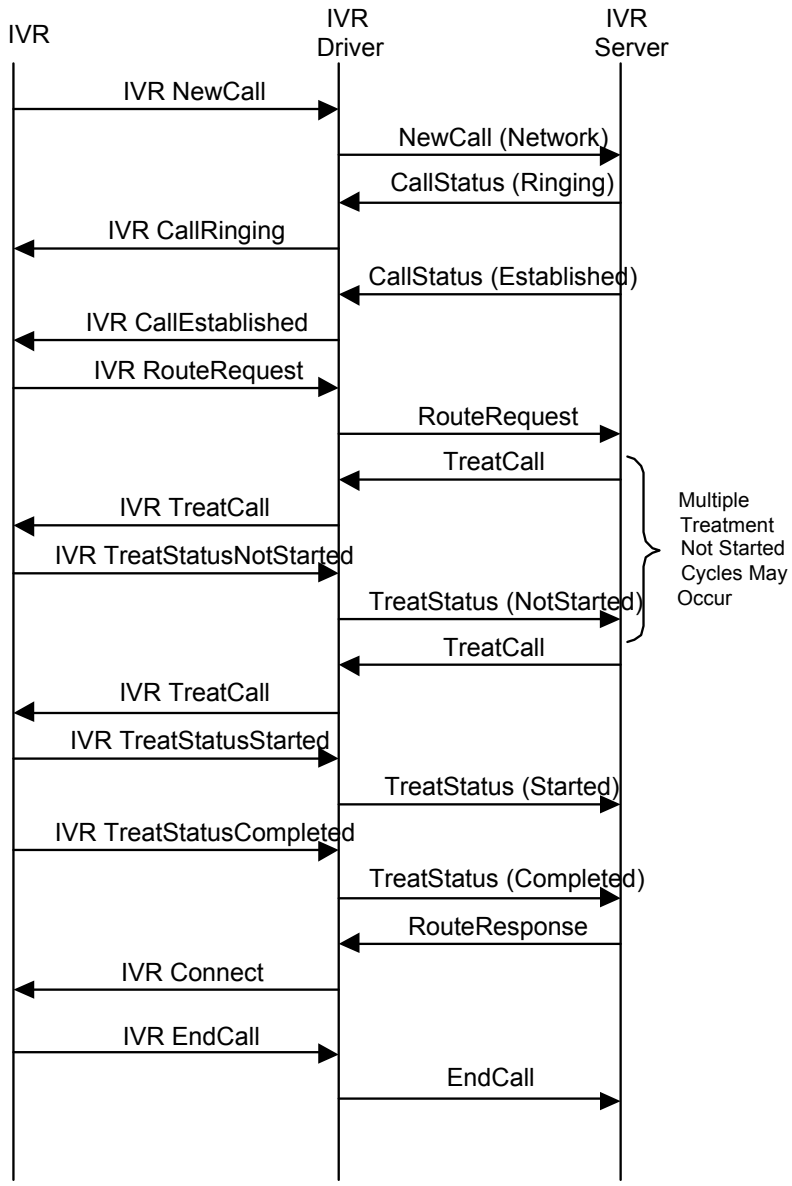

Figure 11: Call Treatment Failed Call Flow

Call Treatment Interrupted

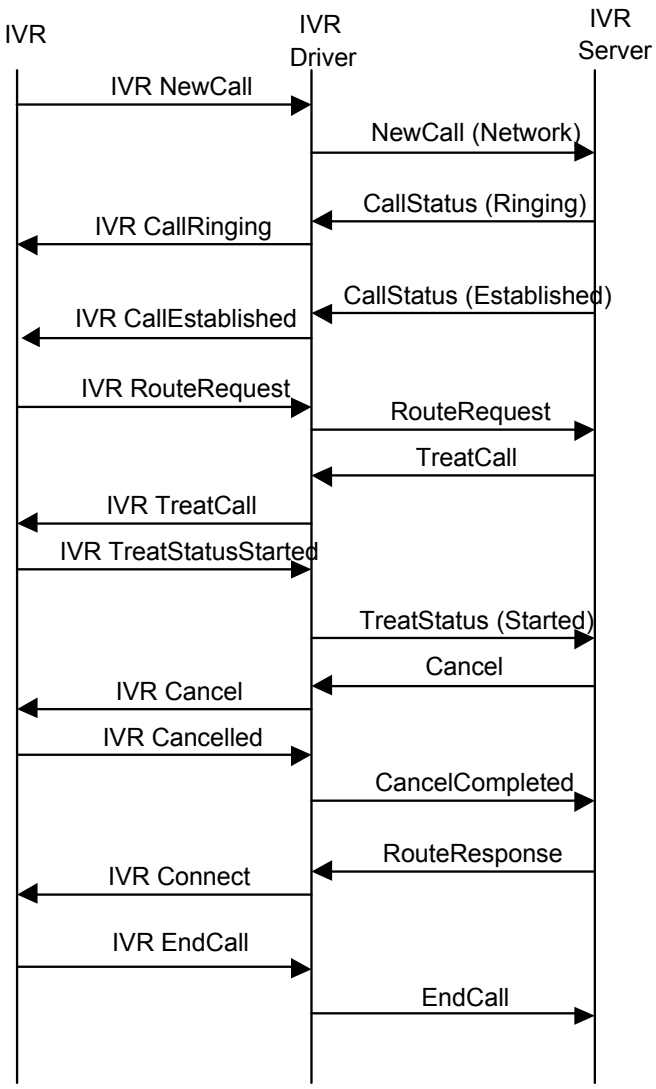

Figure 12: Interrupted Call Treatment

The command to cancel the call treatment is forwarded from the Genesys Framework by IVR Server.

MakeCall Call Flow

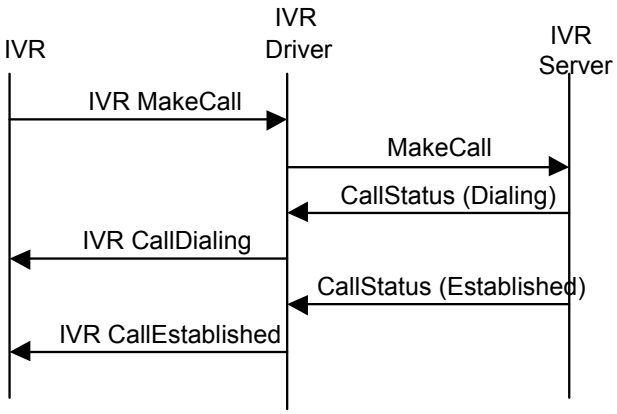

Figure 13: MakeCall Operation

Note: The `MakeCall` call flow is relevant to the behind mode only.

See "MakeCall Operation" on page 109 for sample XML messages and comments on the IVR driver/IVR Server segment of this call flow.

MakeCall (Busy)

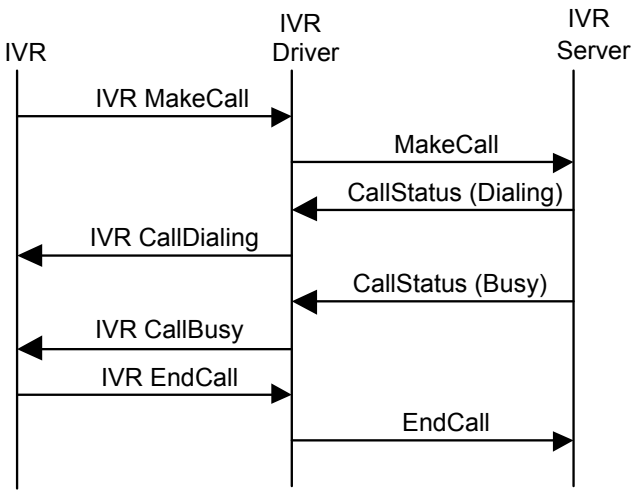

Figure 14: MakeCall (Busy) Call Flow

Conference Call Flow Diagrams

The following call flow diagrams illustrate several scenarios involving conferenced calls.

One-Step Conference

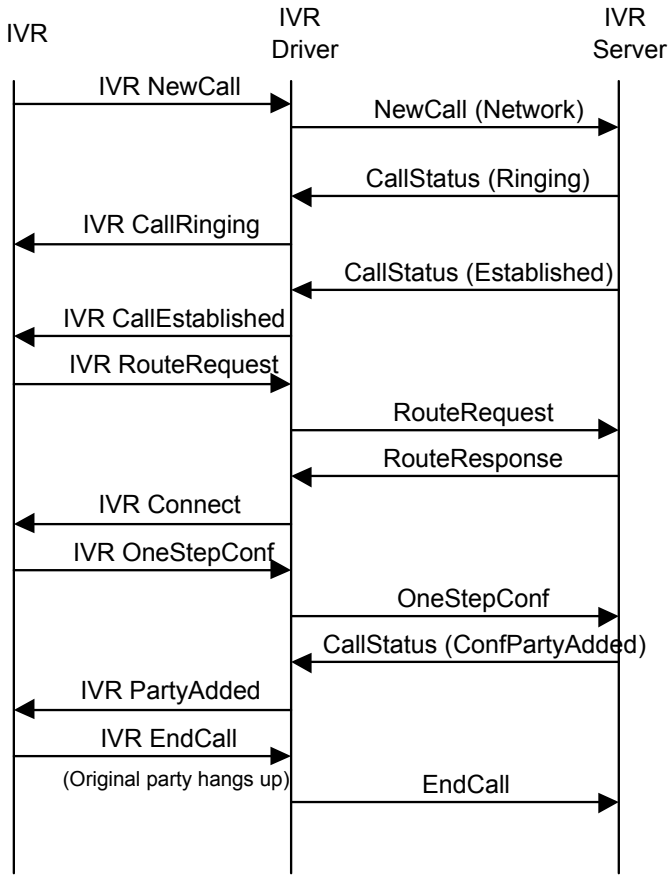

Figure 15: Call Flow for a One-Step Conference

See "One-Step Conference Operation" on page 111 for sample XML messages and comments on the IVR driver/IVR Server segment of this 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 original call is retrieved without any input from the IVR.

One-Step Conference, Scenario 2

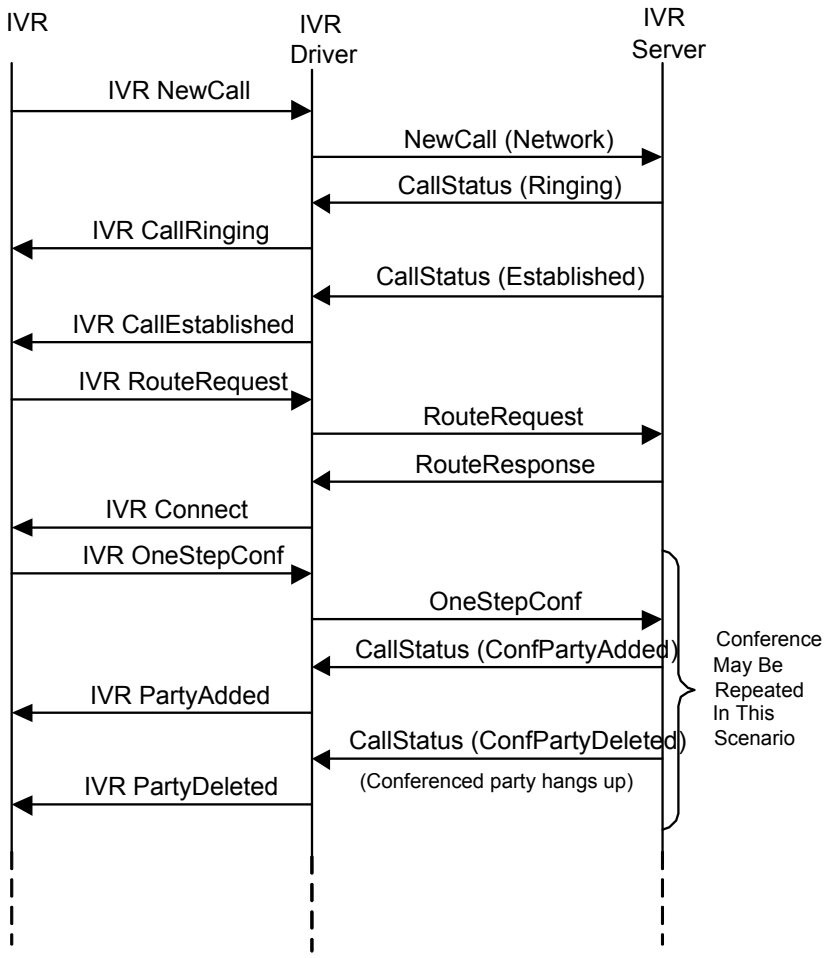

Figure 16: One-Step Conference with Alternative Disconnect Scenario

Conference Consult Call

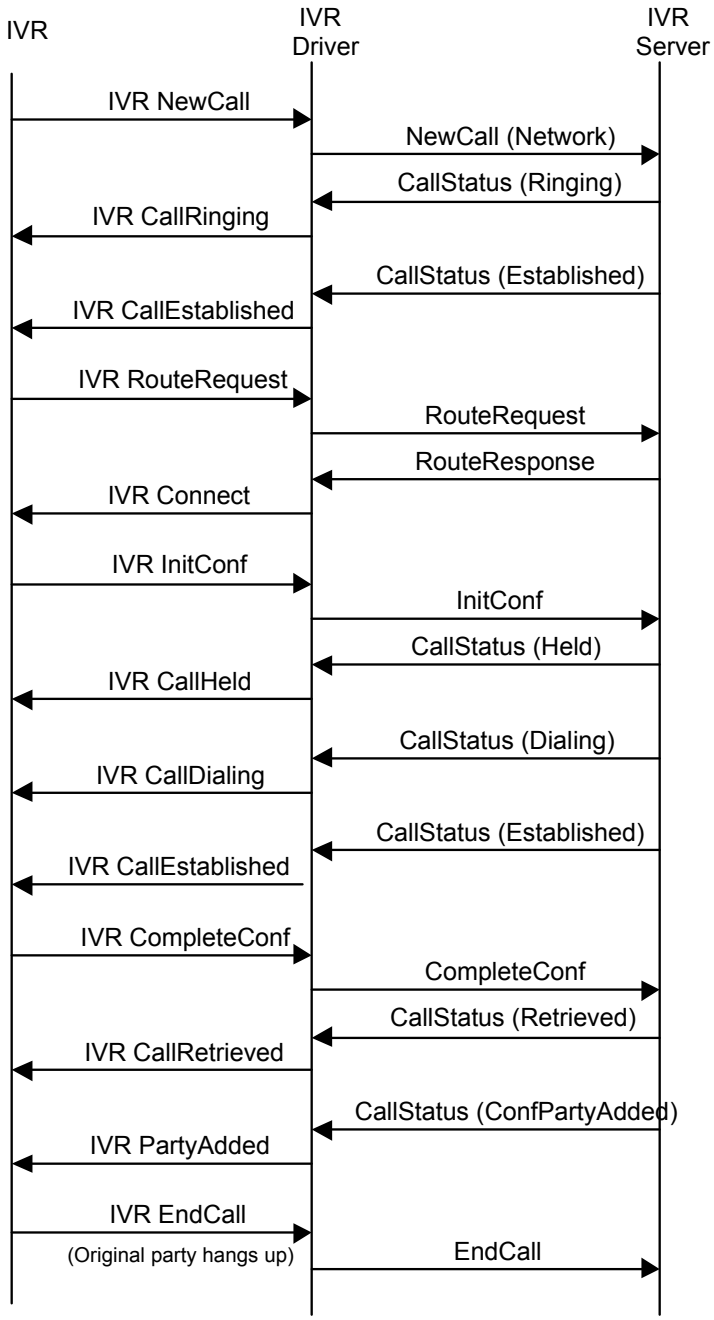

Figure 17: Call Flow for Conference Consult Call

See “Conference Consult Operation” on page 113 for sample XML messages and comments on the IVR driver/IVR Server segment of this call flow.

Conference Consult Call, Scenario 2

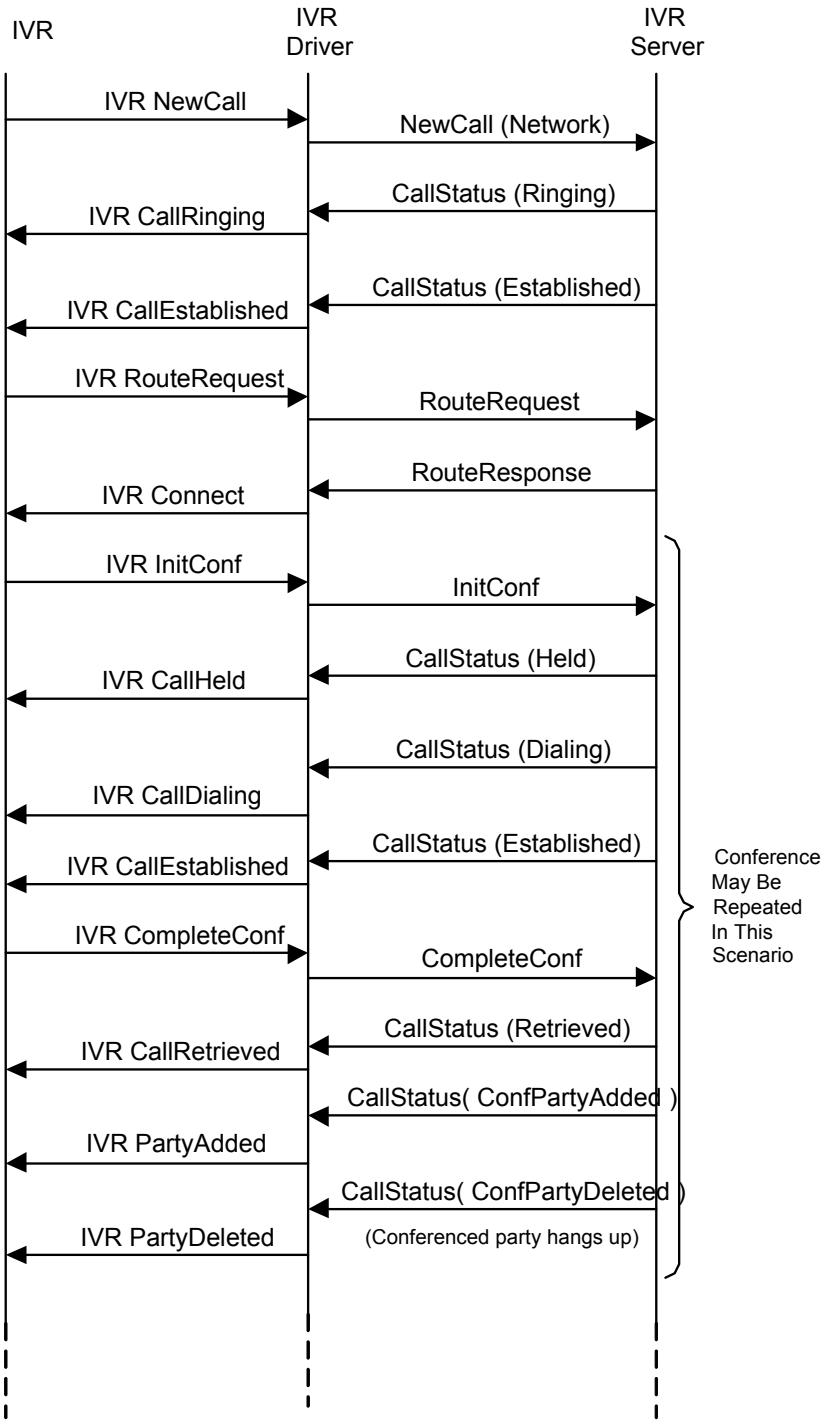

Figure 18: Conference Consult Call Alternative Scenario

Conference Consult Call (Busy)

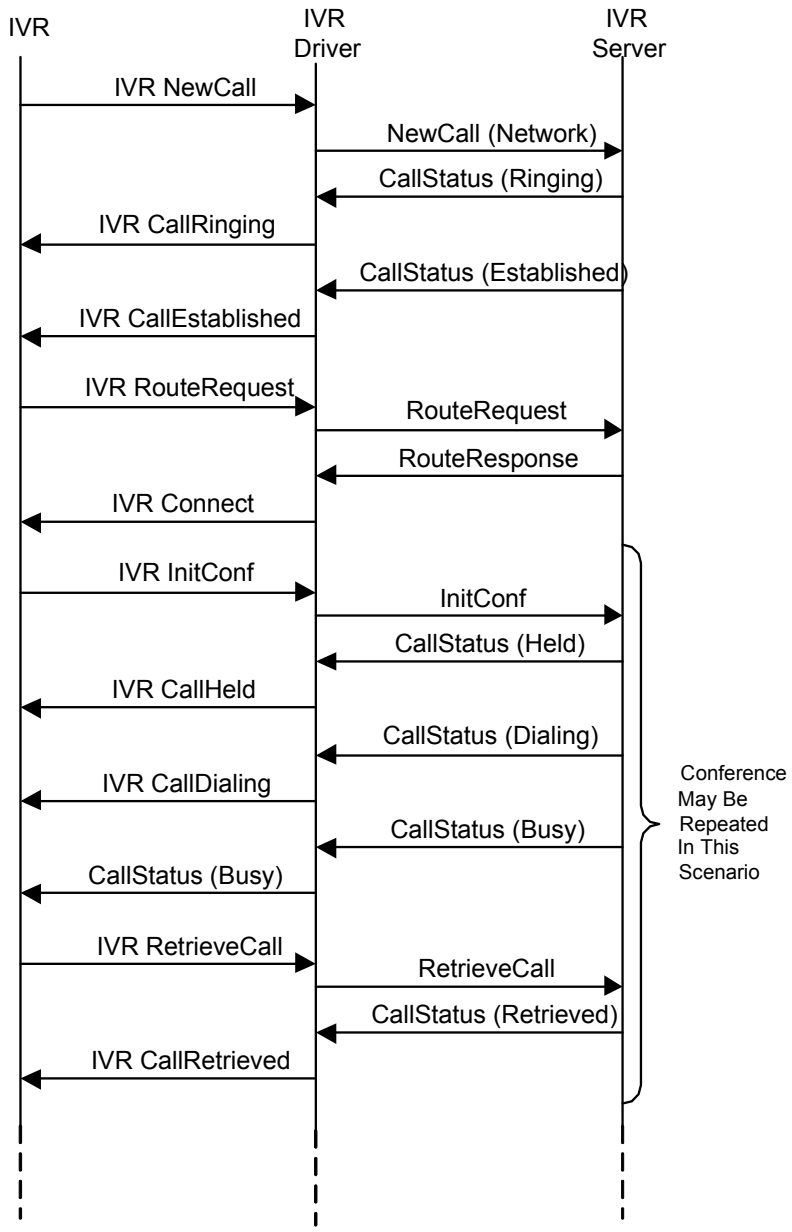

Figure 19: Conference Consult Call, Line Busy

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)” on page 54.

Conference Consult Call (Failed)

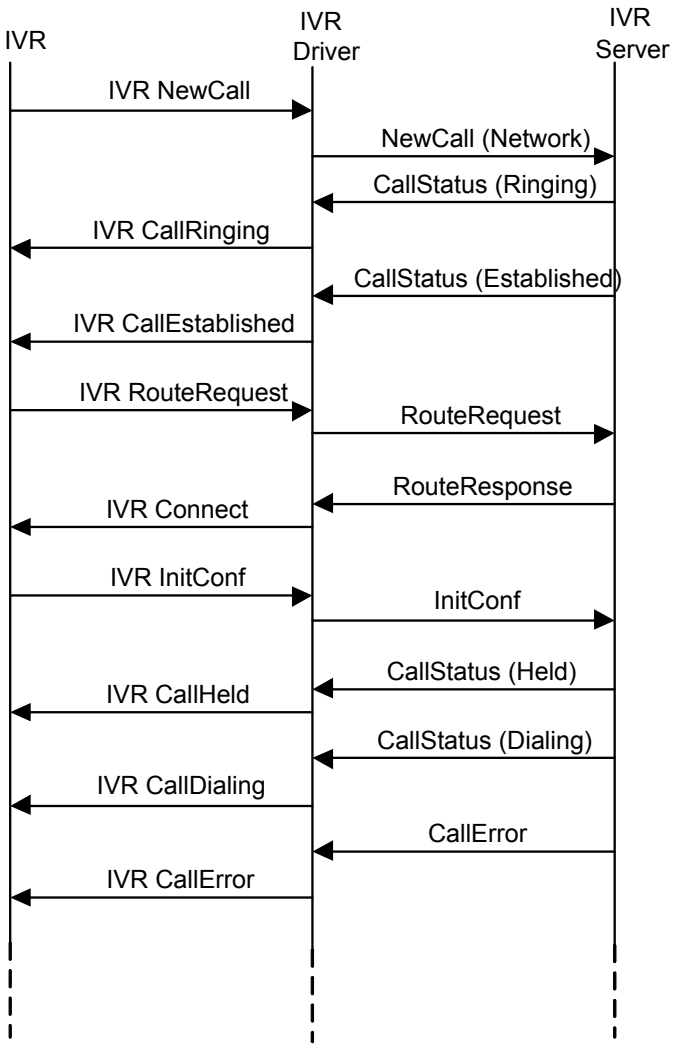

Figure 20: Conference Consult Call Failed Call Flow

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.

Note: 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

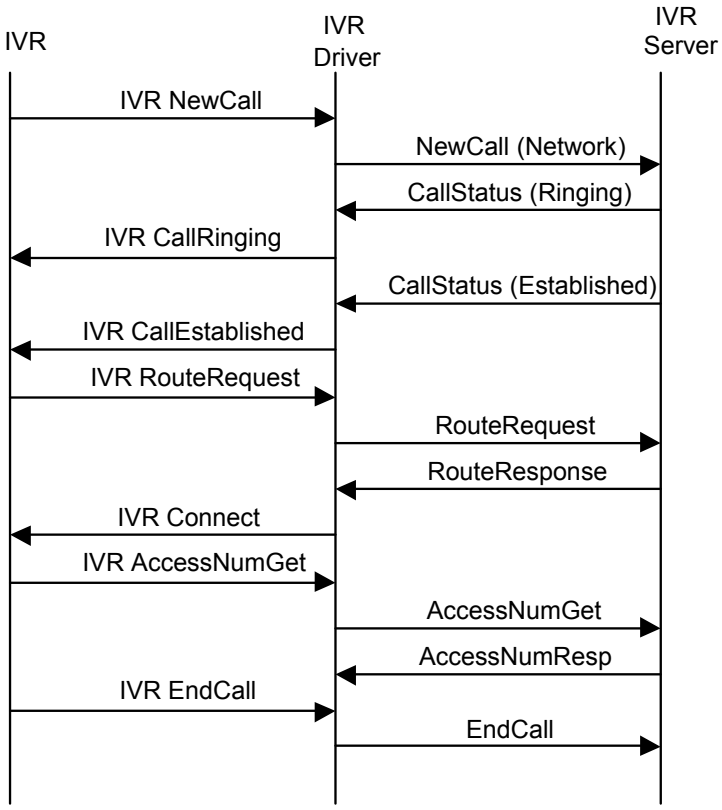

Figure 21: Transfer to a Remote Site

See "Transfer to Remote Site Operation" on page 118 for sample XML messages and comments on the IVR driver/IVR Server segment of this call flow.

Single-Step Transfer

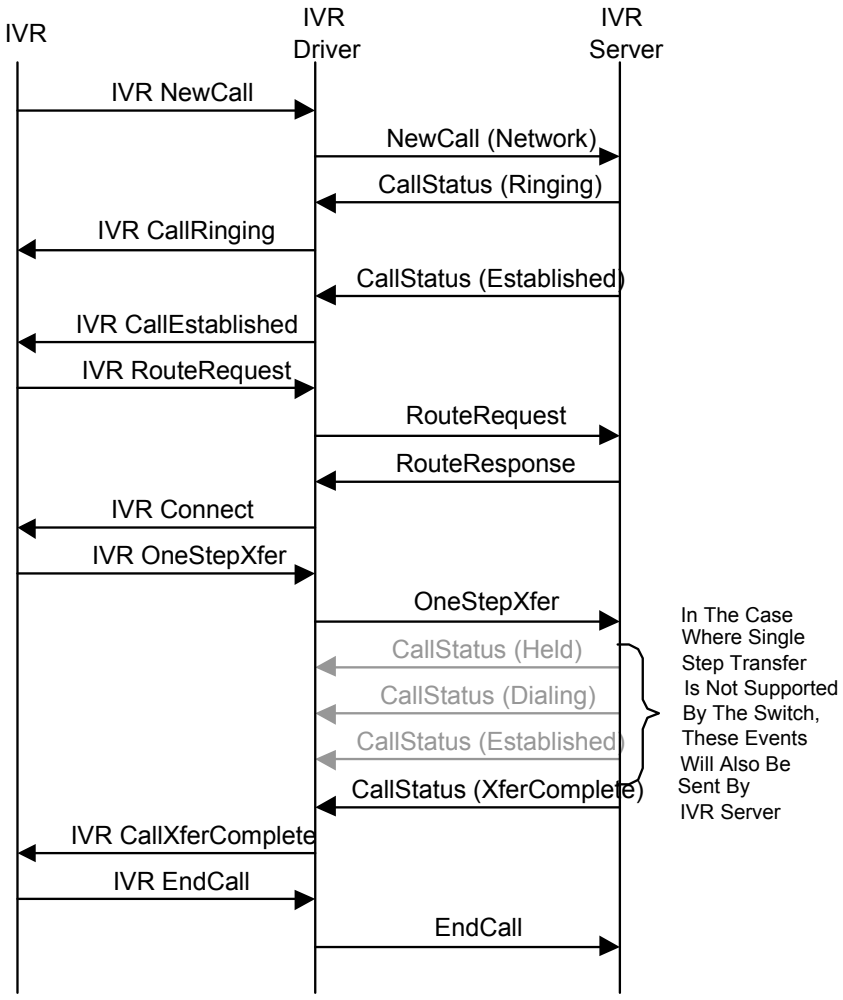

Figure 22: Single-Step Transfer

See "One-Step Transfer Operation" on page 120 for sample XML messages and comments on the IVR driver/IVR Server segment of this call flow.

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

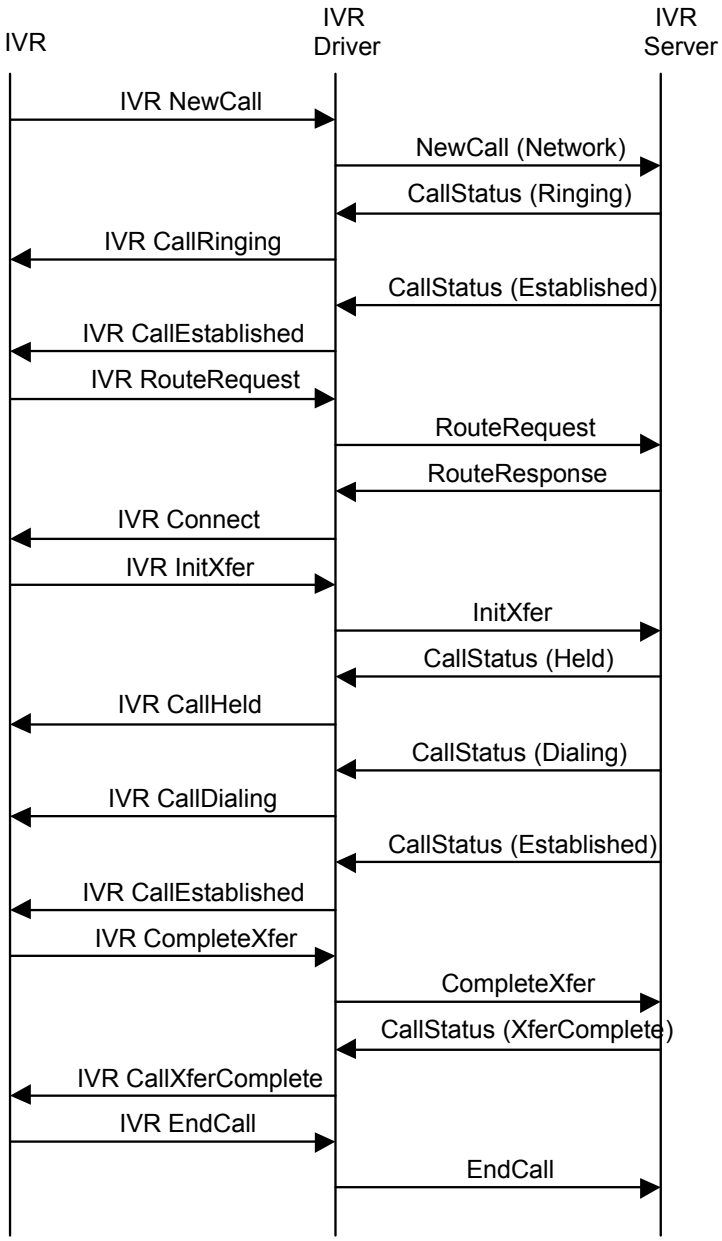

Figure 23: Call Flow for a Transfer Consult Call

See “Transfer Consult Operation” on page 122 for sample XML messages and comments on the IVR driver/IVR Server segment of this call flow.

Transfer Consult Call (Busy)

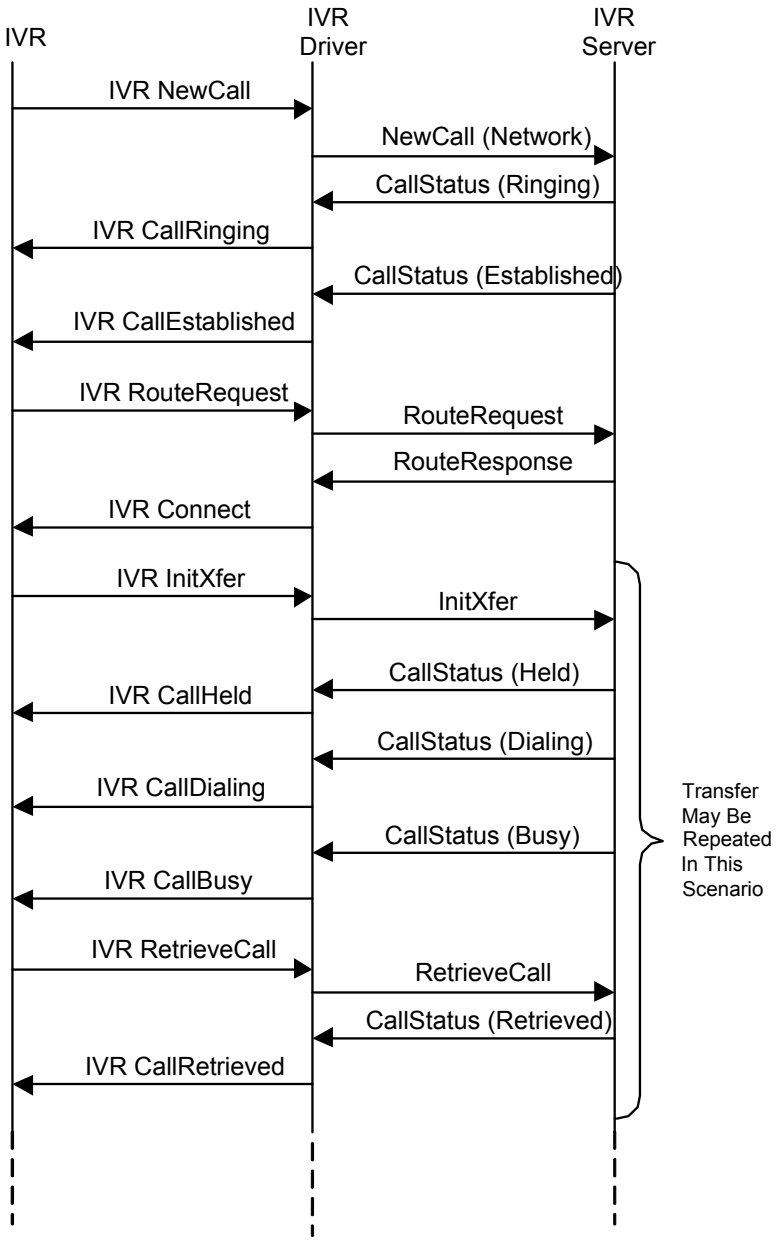

Figure 24: Transfer Consult Call, Line Busy

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)" on page 59.

Transfer Consult Call (Failed)

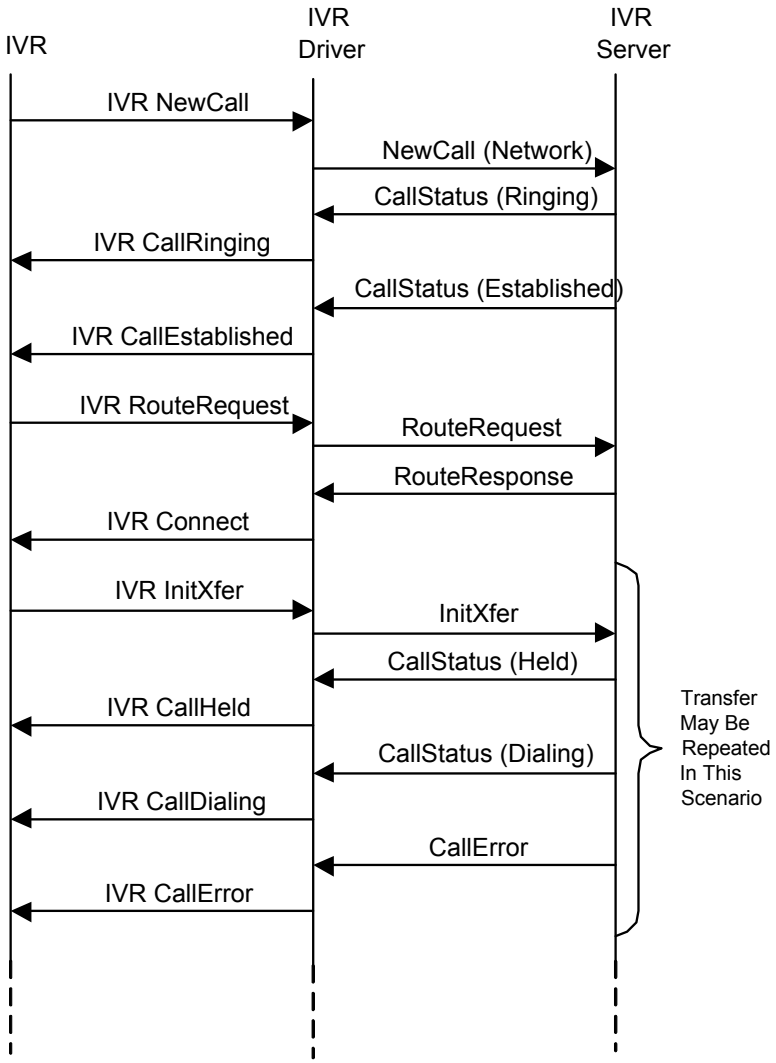

Figure 25: Transfer of Consult Call Failed

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.

Note: 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.

Chapter

5 IVR XML Protocol Messages and Parameters

This chapter presents detailed explanations of the messages and parameters used by the Genesys IVR XML protocol in a standard deployment situation. This chapter contains these sections:

Overview

The messages in this chapter are defined in version 3.0 of the `IServer.dtd` file. For a complete text of the DTD, see Appendix, "The IVR Server DTD," on page 173.

Important Message Constraints

CallId The value of the `CallId` parameter can be any valid character string and is assigned by the IVR driver at the time of the initial `NewCall` or `MakeCall` request is made. This `CallId` must be included in every subsequent message for this call.

The `CallId` field used in the `Login` message will be used for all server monitoring events generated by IVR Server. Any control messages sent by the IVR must use the `CallId` supplied in `Login` as well. Messages not using this `CallId` will result in error messages.

See the DTD (in Appendix, "The IVR Server DTD," on page 173) for details on how `CallId` fits into the message structure.

Note: The `CallId` parameter must be unique for every concurrently active call handled by an IVR Server application. This means not only that each individual client application has to generate unique `CallIds`, but if multiple clients connect to the same IVR Server application, those clients must ensure that they do not use the same `CallIds`.

Deprecated Parameters

The version 1.0 parameters `UserData` and `Extensions` are still supported, but are deprecated. The `UDataEx` and `ExtnsEx` parameters should be used instead, and are supported in DTD version 2.0 and higher.

Key/Data Pairs

The key in a key/data pair cannot include the character ":". An error message appears when the user tries to retrieve data identified using this character in the key.

Special Character Encoding

The characters `<`, `>`, `&`, `"`, and `'` must all be coded in escape form for the parser to interpret them correctly. The escaped forms are:

Table 5: Escape Form for Certain Characters

Character	Escape Form
<	<
>	>
&	&
"	"
'	'

Note: The messaging protocol used between IVR Server and IVR Clients (for instance, Genesys IVR Drivers, Genesys Voice Platform (GVP), and custom-built SDK clients) is XML, and certain character codes are not valid in an XML document. Characters with a value less than hexadecimal number `0x20` are not valid, with the exception of the characters `0x09`, `0x0A`, and `0x0D`. These exceptions correspond to the ASCII control characters `TAB`, `LINE FEED`, and `CARRIAGE RETURN`. Application user-data keys or values should not contain any of the disallowed character codes.

General Messages

These messages include login, logging, and reset messages.

LoginReq

Sent by the IVR to the IVR Server to initiate a session and authenticate user access to the IVR Server. Except when working in Network mode, this request is required for the client to interact with the IVR Server. (While using login in network mode is not recommended, it is allowed for compatibility with the previous XML interface (GenSpec XML) if one is sent. However, it must not specify `ReportStatus=true`.) `LoginReq` is the first XML request the client must send to the IVR Server.

Note: Without sending `LoginReq` and receiving a successful `LoginResp` (see "LoginResp" on page 64), the client cannot send any further XML requests to the IVR Server.

The value for the `ClientName` parameter is the name given to the IVR `Application` object in Configuration Manager.

Set the optional `ReportStatus` parameter to `true` to indicate that the login response message (`LoginResp`) for this request should include its `Status` parameter. You can also use the `Status` parameter to determine if the IVR Server startup initialization is still in progress (and not able to process calls on all ports). To determine when initialization is complete, have your application periodically send login requests until the status result is `OK`.

Extending the agent login mechanism to include a client-driven approach makes it necessary to generate certain unsolicited events. T-Server messages that were previously of no interest to a connected IVR will now be crucial to proper agent state management. You must subscribe to events such as `EventLinkConnected` and `EventLinkDisconnected` to maintain backwards compatibility. The `ServerMonitor` parameter will extend the existing login message for subscribing to these events.

Set the optional `ServerMonitor` parameter to `set` to indicate that a particular client is interested in significant remote events. If the client is no longer interested in this information, resending a `Login` message with `ServerMonitor` set to `clear` will remove the feature. If this field is not set, the default is considered to be `clear`. The `ServerMonitor` parameter is respected only if `Version` is set to `4.0`.

Note: Login in IVR Server cannot fail. It can also be repeated without negative effects.

Place required configuration information in the `data transport` section of the IVR `Application` object in Configuration Manager. In that case, the information is returned in the `ConfigOptions` section of the `LoginResp` message.

See Table 6 for a complete list of message parameters.

Table 6: LoginReq Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
LoginReq	IVR to IVR Server	Version	1.0 2.0 3.0 4.0	Required
		ClientName		Required
		ReportStatus	true false	Optional **Note:** When using Network mode, do not set this parameter to `true`.
		ServerMonitor	set clear	Optional

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. (See Table 7 for a complete list of message parameters.)

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.

Clients can use the value of the Status parameter to detect IVR Server's initialization status. If it is initializing, clients can then periodically retry LoginReq until IVR Server initialization is complete.

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.

Table 7: LoginResp Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
LoginResp	IVR Server to IVR	IServerVersion		Required
		Result	Success InvalidProtocolVersion	Required
		ConfigOptions		Optional
		Status	NoSuchClient InitInProgress OK	Optional

LogMsg

This message from the IVR to the IVR Server allows you to write a message to a log file. You can specify the log type and the desired log message.

This file can be local or on the IVR Server. Set the log location in the Data Options Transport section of the IVR Server Application in Configuration Manager.

See Table 8 for a complete list of message parameters.

Table 8: LogMsg Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
LogMsg	IVR to IVR Server	MsgType	Standard Trace Debug	Required
		Msg		Required

Reset

This message is not yet implemented. It is reserved for future use.

See Table 9 for a complete list of message parameters.

Table 9: Reset Message

Message	Direction	Parameter Name	Optional/ Required
Reset	IVR to IVR Server	ExtnsEx	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.

See Table 10 for a complete list of message parameters.

Table 10: MonitorInfo Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
MonitorInfo	IVR Server to IVR	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 are shown in Table 11.

See Table 12 for a complete list of message parameters.

Table 11: Server Status Events

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

Table 12: Server Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
Server	IVR Server to IVR	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.

See Table 13 for a complete list of message parameters.

Table 13: Port Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
Port	IVR Server to IVR	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. Table 14 shows the relationship between T-Library events and XML messaging.

Table 14: 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`.

See Table 15 for a complete list of message parameters.

Table 15: Agent Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
Agent	IVR Server to IVR	PortNum		Required
		Status	LoggedIn LoggedOut Ready NotReady Unknown	Required

AgentQuery

The client can, at any time, request the current agent state. This will trigger the generation of a `MonitorInfo` (`Agent` subtype) message. Functionally, this triggers a `TQueryAddress` on behalf of the IVR. The values that can be returned are taken from the `AgentStatus` extension to `EventAddressInfo` and are translated in Table 16.

See Table 17 on page 70 for a complete list of message parameters.

Table 16: Agent Status Values

AgentStatus	Value	XML Status
UNKNOWN	< 0	Unknown
LOGGED_OUT	0	Unknown
LOGGED_IN	1	LoggedIn
READY	2	Ready
NOT_READY	3	NotReady
ACW	4	NotReady
WALK_AWAY	5	NotReady
Any other value		Unknown

Table 17: AgentQuery Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
AgentQuery	IVR to IVR Server	ReqId		Required
		PortNum		Required

AgentLogin

This message is sent when the IVR wishes to log an agent in. This message converts to a `TAgentLogin` message. Generally this message will only be acceptable to the switch when the current agent state is logged out.

See Table 18 for a complete list of message parameters.

Table 18: AgentLogin Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
AgentLogin	IVR to IVR Server	ReqId		Required
		PortNum		Required
		Queue		Required
		Agent Id		Required
		Password		Required

AgentLogout

This message is sent when the IVR wishes to log an agent out. This message converts to a `TAgentLogout` message. Generally this message will only be acceptable to the switch when the current agent state is not logged out, though behavior can vary from switch to switch.

See Table 19 for a complete list of message parameters.

Table 19: AgentLogout Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
AgentLogout	IVR to IVR Server	ReqId		Required
		PortNum		Required
		Queue		Optional

AgentReady

This message is sent to set an existing logged in port to the ready state. The optional `WorkMode` parameter is converted to a corresponding `AttributeWorkMode` in the `TAgentSetReady` message.

See Table 20 for a complete list of message parameters.

Table 20: AgentReadyMessage

Message	Direction	Parameter		Optional/ Required
		Name	Value	
AgentReady	IVR to IVR Server	ReqId		Required
		PortNum		Required
		Queue		Required
		WorkMode	AutoIn ManualIn Unknown	Optional

AgentNotReady

This message is sent to set an existing logged in port to the not ready state. The optional `WorkMode` is converted to a corresponding `AttributeWorkMode` in the `TAgentSetNotReady` message.

See Table 21 for a complete list of message parameters.

Table 21: AgentNotReady Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
AgentNotReady	IVR to IVR Server	ReqId		Required
		PortNum		Required
		Queue		Required
		WorkMode	AutoIn ManualIn Unknown	Optional

FlowControl

This message is sent based on the setting of the `flow-control` option in the IVR Server's `Application` object properties. The message indicates the current setting for flow control, and is returned at login or when the value changes. When the status of flow control is `on`, new calls may be rejected depending on the state of the driver or IVR Server. The call ID used comes from the corresponding `LoginReq`.

Note: For the XML-based client to receive and handle this flow control message, it must log in using the 4.0 version of the `IServer.dtd` file.

See Table 22 for the message parameters.

Table 22: FlowControl Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
FlowControl	IVR Server to IVR	Status	on off	Required

New Call and Call Routing Messages

These messages are used to start a call, route it, confirm the connection or indicate failure to connect, and end the call.

NewCall

This message is sent by the IVR to the IVR Server to indicate the start of a new call.

See Table 23 for a complete list of message parameters.

Table 23: New Call Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
NewCall	IVR to IVR Server	CalledNum[a,b]		Required
		CallControlMode[c]	Genesys Network (MakeCall)	Required
		Version	1.0 2.0 3.0 4.0	Required
		ANIRestriction	CLIP—(Calling Line Identification Presentation) CLIR—(Calling Line Identification Restriction)	Reserved for Future Use
		DNIS		Optional
		ANI		Optional
		UDataEx		Optional
		ExtnsEx		Optional

a. Recall that the `CalledNum` parameter is the IVR port number of the call. (See the `CalledNum` definition on page 40.)

b. It is the responsibility of the XML driver to ensure that there is only one call instance at a time per `CalledNum`. So, once a `CalledNum` is in use, it must be ended (with `EndCall`) prior to using the same `CalledNum` (which is an IVR Port) again.

c. When `CallControlMode` is set to `Genesys`, Routing Server (URS) controls this call. When set to `Network`, the IVR controls the call. In the latter case, the IVR can request to have the call routed by URS by sending a `RouteRequest` message. The value `MakeCall` is for backwards compatibility purposes only. (It allows the premise T-Server, release 6.1, to initiate calls.)

RouteRequest

This message is sent by the IVR to the IVR Server to request that the call be routed by Genesys URS. This message can only be used when `CallControlMode` is set to `Network` in the `NewCall` message. It indicates that the call has been processed by the IVR and invokes a routing strategy. Note that the call can be parked prior to routing if necessary.

Note: For In-Front and Behind mode IVRs, if the Router does not respond prior to the timeout, this message results in `RouteResponse(RouteType=Default)`. (See "RouteResponse" for details.) For IVRs in Network mode, a Router timeout results in `EndCall(EndCause=Timeout)`. (See "EndCall" on page 75 for details. This Network mode behavior provides compatibility with the GenSpec XML NTS.)

See Table 24 for a complete list of message parameters.

Table 24: RouteRequest Message

Message	Direction	Parameter Name	Optional/Required
RouteRequest	IVR to IVR Server	RouteDn (DN where the routing strategy is located.)	Required for IVR in front and IVR behind. Ignored for IVR with Network T-Server.
		CED	Optional
		UDataEx	Optional
		ExtnsEx	Optional

RouteResponse

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

See Table 25 for a complete list of message parameters.

Table 25: RouteResponse Message

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

EndCall

This message is sent by either the IVR or the IVR Server response to a `NewCall` message.

Note: If the IVR (driver) generates `EndCall` while it still has outstanding requests, the behavior of IVR Server in replying to those requests is undefined. In some cases replies may be sent, but not in all cases. No further call-related messages may arrive after the `EndCall` message.

If you issue `EndCall` with `GCTIActiveRelease` set to `false`, the active call on the IVR port is not terminated (if it is still on the port). Issuing `EndCall` with this parameter not set (which is the default behavior), also causes the call to be cleared.

See Table 26 on page 76 for a complete list of message parameters.

Table 26: EndCall Message

Message	Direction	Parameter		Optional/Required
		Name	Value	
EndCall	IVR Server to IVR *OR* IVR to IVR Server	EndCause	Normal Abandoned Resources FeatureNotSupported InvalidVersion InvalidStateTransition ServerPaused Timeout	Required
		UDataEx		Optional (Used for Network IVR)
		ExtnsEx		Optional (Used for Network IVR)
		GCTIActive-Release	true false	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.

See Table 27 for a complete list of message parameters.

Table 27: TreatCall Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
TreatCall	IVR Server to IVR	Type	Busy CancelCall CollectDigits DeleteAnnounce FasyBusy IVR Music PlayAnnounce PlayAnnounceAndDigits PlayApplication RAN RecordAnnouce RingBack SetDefaultRoute Silence TextToSpeech TextToSpeechAndDigits VerifyDigits	Required
		ExtnsEx	AttributeExtensions	Optional
		Parameters	AttributeTreatmentParms	Optional

TreatStatus

This message is sent by the IVR to the IVR Server to indicate what the call treatment process requested by the IVR Server is doing.

See Table 28 for a complete list of message parameters.

Table 28: TreatStatus

Message	Direction	Parameter		Optional/ Required
		Name	Value	
TreatStatus	IVR to IVR Server	CallId		Required
		Status	Started NotStarted—(Indicates the the call treatment did not start properly) Completed	Required
		UDataEx		Optional
		ExtnsEx		Optional
		CED		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.

See Table 29 for a complete list of message parameters.

Table 29: Cancel Message

Message	Direction	Parameter Name	Optional/ Required
Cancel	IVR Server to IVR		

CancelCompleted

This message is sent by the IVR to the IVR Server to indicate that the call treatment requested by the IVR Server has been canceled.

This message has no parameters.

Table 30: CancelCompleted Message

Message	Direction	Parameter Name	Optional/ Required
CancelCompleted	IVR to IVR Server		

External Routing Messages

These messages are used to prepare a call for an inter-switch transfer. They make it possible for caller data to be transferred from one switch/T-Server to another at a different site.

AccessNumGet

This message is sent by the IVR to the IVR Server to request that the call be routed to a remote site. The `XRouteType` parameter is used to select the type of routing required.

Note: This functionality is not supported when IVR Server operates in Network Mode.
When IVR Server operates in In-Front or Behind modes, in order to communicate with T-Server, IVR Server translates `AccessNumGet` into the T-Library function call `TGetAccessNumber()`. In these modes, IVR Server acts as a T-Library client, and so is able to make this request of T-Server. When operating in Network mode, however, IVR Server does not act as a T-Library client, and so has no way to generate the `AccessNumGet` request.

See Table 31 on page 80 for a complete list of message parameters.

Table 31: AccessNumGet Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
AccessNumGet	IVR to IVR Server	DestDN		Required
		Location (Indicates the switch to which to call is transferred.)		Required
		XRouteType	Default Route Reroute Direct DirectAni DirectNoToken DirectAniDnis DirectUUI DirectDigits DnisPool	Optional (`Default` is the default)
		UUI_Number		Optional
		UDataEx		Optional
		ExtnsEx		Optional

AccessNumCancel

This message is sent by the IVR to the IVR Server to request that the previous `AccessNumGet` request be canceled. There are no parameters in this message.

Table 32: AccessNumCancel Message

Message	Direction	Parameter Name	Optional/ Required
AccessNumCancel	IVR to IVR Server		

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`.

See Table 33 for a complete list of message parameters.

Table 33: AccessNumResp Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
AccessNumResp	IVR Server to IVR	Action	Get Cancel	Required
		Result	Success Failure	Required
		AccessNum		Optional

Transfer/Conferencing Messages

These messages are used to control call transfers and conferencing.

OneStepXfer

This message is sent by the IVR to the IVR Server to request that the call be immediately transferred to another agent. This message is present in Behind mode only.

See Table 34 for a complete list of message parameters.

Table 34: One-Step Transfer Message

Message	Direction	Parameter Name	Optional/ Required
OneStepXfer	IVR to IVR Server	DestDN	Required
		Location (Indicates the switch to which to call is transferred)	Optional
		UDataEx	Optional
		ExtnsEx	Optional

OneStepConf

This message is sent by the IVR to the IVR Server to request that another agent be immediately conferenced into the call.

See Table 35 for a complete list of message parameters.

Table 35: OneStepConf Message

Message	Direction	Parameter Name	Optional/ Required
OneStepConf	IVR to IVR Server	DestDN	Required
		Location (Indicates the switch to which to call is transferred)	Optional
		UDataEx	Optional
		ExtnsEx	Optional

InitXfer

This message is sent by the IVR to the IVR Server to request that the call be transferred to another agent.

See Table 36 for a complete list of message parameters.

Table 36: InitXfer Message

Message	Direction	Parameter Name	Optional/ Required
InitXfer	IVR to IVR Server	DestDN	Required
		Location (Indicates the switch to which to call is transferred)	Optional
		UDataEx	Optional
		ExtnsEx	Optional

InitConf

This message is sent by the IVR to the IVR Server to request that another agent be conferenced into the call.

See Table 37 for a complete list of message parameters.

Table 37: InitConf Message

Message	Direction	Parameter Name	Optional/ Required
InitConf	IVR to IVR Server	DestDN	Required
		Location (Indicates the switch to which to call is transferred)	Optional
		UDataEx	Optional
		ExtnsEx	Optional

CompleteXfer

This message is sent by the IVR to the IVR Server to indicate that the transfer has been completed.

See Table 38 for a complete list of message parameters.

Table 38: CompleteXfer Message

Message	Direction	Parameter Name	Optional/ Required
CompleteXfer	IVR to IVR Server	ExtnsEx	Optional

CompleteConf

This message is sent by the IVR to the IVR Server to indicate that the conference call has been set up.

See Table 39 for a complete list of message parameters.

Table 39: CompleteConf Message

Message	Direction	Parameter Name	Optional/ Required
CompleteConf	IVR to IVR Server	ExtnsEx	Optional

RetrieveCall

This message is sent by the IVR to the IVR Server to request that the original call be retrieved from hold.

See Table 40 for a complete list of message parameters.

Table 40: RetrieveCall Message

Message	Direction	Parameter Name	Optional/ Required
RetrieveCall	IVR to IVR Server	ExtnsEx	Optional

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.

See Table 41 for a complete list of message parameters.

Table 41: CallStatus Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
CallStatus	IVR Server to IVR	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.

See Table 42 on page 86 for a complete list of message parameters.

Table 42: CallError Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
CallError	IVR Server to IVR	FailedReq	Unknown NoSuchCall OneStepXfer OneStepConf InitConf CompleteConf InitXfer CompleteXfer RetrieveCall MakeCall AgentControl NotAllowed	Required
		TLibErrCode		Optional
		ReqId		Optional

Call Information Messages

These messages request data attached to the call and return the corresponding response.

CallInfoReq

See “CallInfoResp”, below, for the information this request can produce.

Table 43: CallInfoReq Message

Message	Direction	Parameter		Optional/ Required
	Direction	Name	Value	
CallInfoReq	IVR to IVR Server	ReportUUID	true false	Optional
		ReportThirdPartyDN	true false	Optional

CallInfoResp

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

Note: 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 the *Voice Platform SDK 8.x .NET* or *Java API Reference* for details.

See Table 44 for a complete list of message parameters.

Table 44: CallInfoResp Message

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

Statistics Messages

The statistics messages enable you to request and 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. For instructions on configuring statistics, see “Configuring Stat Server Statistics” on page 42.

PeekStatReq

The `PeekStatReq` message returns the current values for the requested statistics.

See Table 45 for a complete list of message parameters.

Table 45: PeekStatReq Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
PeekStatReq	IVR to IVR Server	RequestId		Required
		StatName		Required

GetStatReq

The `GetStatReq` message returns a full report on the requested statistics for the specified objects (queue, route point, group of queues).

See Table 46 for a complete list of message parameters.

Table 46: GetStatReq Message

Message	Direction	Parameter Name	Optional/ Required
GetStatReq	IVR to IVR Server	RequestId	Required
		ServerName	Required
		StatType	Required
		ObjectId	Required
		ObjectType	Required

StatResp

Supplies the response to the `PeekStatReq` and `GetStatReq` messages.

See Table 47 for a complete list of message parameters.

Table 47: StatResp Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
StatResp	IVR Server to IVR	RequestId		Required
		ResultCode	Success NoSuchStat MiscError	Required
		Result		Optional

User Data Messages

These messages enable you to access and control data about the actions performed by callers.

UDataGet

Requests the values for existing keys. The keys to retrieve should be entered in a colon-separated list. This use of colons as separators is the reason that colons cannot be used as a part of a key name.

See Table 48 for a complete list of message parameters.

Table 48: UDataGet Message

Message	Direction	Parameter Name	Optional/ Required
UDataGet	IVR to IVR Server	RequestId	Required
		Keys	Required

UDataGetAll

Request the values for all keys present in the call's user data. The keys to retrieve should be entered in a colon-separated list. This use of colons as separators is the reason that colons cannot be used as a part of a key name.

See Table 49 for a complete list of message parameters.

Table 49: UDataGetAll Message

Message	Direction	Parameter Name	Optional/ Required
UDataGetAll	IVR to IVR Server	RequestId	Required

UDataSet

This message enables you to add or change (replace) user data keys.
See Table 50 on page 90 for a complete list of message parameters.

Table 50: UDataSet Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
UDataSet	IVR to IVR Server	RequestId		Required
		Action	Add Replace	Required
		UDataEx		Optional

UDataDel

This message allows you to delete one or all user data keys.
See Table 51 for a complete list of message parameters.

Table 51: UDataDel Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
UDataDel	IVR to IVR Server	RequestId		Required
		Action	DeleteAll DeleteKey	Required
		Key		Optional

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.

See Table 52 for a complete list of message parameters.

Table 52: UDataResp Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
UDataResp	IVR Server to IVR	RequestId		Required
		Result	Success NoSuchCall NoMatch FeatureNotSupported MiscError	Required
		UDataEx		Optional

Outbound Messages

DialOutRegistry

Sent from the IVR to IVR Server, this message controls registrations for outbound DNs. A client may register for one DN (`Command="Add"`), deregister a single DN (`Command="Remove"`) or deregister all DNs (`Command="RemoveAll"`). Other than the case of `RemoveAll`, the DN field is required. See "DialOutRegistryResp" below for proper responses.

See Table 53 for a complete list of message parameters.

Table 53: Dial Out Registry Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
DialOutRegistry	IVR to IVR Server	Command	Add Remove RemoveAll	Required
		DN		Optional

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.

See Table 54 for a complete list of message parameters.

Table 54: Dial Out Registry Resp Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
DialOutRegistryResp	IVR Server to IVR	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.

See Table 55 for a complete list of message parameters.

Table 55: Dial Out Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
DialOut	IVR Server to IVR	OrigNum		Required
		DestNum		Required
		RefID		Required
		TimeToAnswer		Required

DialOutError

Sent from the IVR to IVR Server, this message is sent in response to a `DialOut` message. This indicates that an outbound call could not be dialed for one of the reasons specified. These will be converted to `EventError` towards the T-Library client with `NotSupported` being equivalent to `TERR_UNSUP_OPER`, `NoTrunks` to `TERR_OUT_OF_SERVICE`, and `MiscError` to `TERR_UNKNOWN`.

Note: Unlike all other IVR Server messaging elements, `DialOut` and `DialOutError` do not contain a `CallId` element. In the case of these messages, the IVR Server is initiating the call with the driver. As the driver is responsible for establishing call IDs, the server cannot supply one.

See Table 56 for a complete list of message parameters.

Table 56: Dial Out Error Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
DialOutError	IVR to IVR Server	Error	NotSupported NoTrunks MiscError	Required
		RefID		Required

DialOutInit

Sent in response to a `DialOut` message from IVR Server, the `DialOutInit` message signifies that an outbound call has been dialed. The message provides

a `CallId` field that will be used for the remainder of this call (the `RefID` will no longer be important). The IVR must also provide the route point or IVR port that is dialing the call, depending on the operational mode. The `Version` parameter is used in the same fashion as `NewCall`.

See Table 57 for a complete list of message parameters.

Table 57: Dial Out Init Message

Message	Direction	Parameter		Optional/ Required
		Name	Value	
DialOutInit	IVR to IVR Server	RefID		Required
		Version	2.0 3.0	Optional

Message Parameters

These parameters are composite types based on List and Node. You can create a list containing a string of `UDataEx` or `ExtnsEx` values. See the DTD (in the Appendix, "The IVR Server DTD," on page 173) for details.

UDataEx

The previously used `UserData` tag is considered deprecated, but is supported for backward compatibility. Any message that had `UserData` now accepts the new form, `UDataEx`.

ExtnsEx

The previously used `Extensions` tag is considered deprecated, but is supported for backward compatibility. Any message that had `Extensions` now accepts the new form, `ExtnsEx`.

Genesys

Chapter

6 Using the IVR XML Protocol: Examples

This chapter presents instructions and examples of XML messages showing how to write code for a selection of interaction scenarios. The samples include the following:

GLI Header

Use the code examples in this section to define and insert the GLI header.

Given a pointer to the XML character string to

`send = pSnd`

with

`length = uLen`

a new buffer is allocated with pointer `pNewBuf` which will contain the header information.

The socket write is done with `pNewBuf` and length `newLen`.

Header File

The following C-code example is from a header file that defines the GLI header and `msg` types:

```
#ifdef GLI_VERSION_1
#define GLI_VERSION1
#else
#define GLI_VERSION2
#endif
#
#define GLI_DEFAULT_APP0
static unsigned char   GliHeaderData[ GLI_HEADER_LENGTH ]= {0x0,
0x3, 0x0, 0x2, GLI_VERSION, GLI_DEFAULT_APP};
#define GLI_HEADER_LENGTH     6
```

Adding the Header to the XML Code

The following example is from the `coC_send()` function, which adds the GLI header to the XML code and does the socket write.

```
PBYTE          pNewBuf;
unsigned short newLen;
unsigned short newLenData;
unsigned short htons_newLen;
newLen =  uLen+GLI_HEADER_LENGTH;
if ((pNewBuf=(PBYTE)malloc(newLen)) == NULL)
{ /* Failure. */
prn(WFL_"malloc() for memdup() failure!");
return FALSE; /* Failure. */
}
newLenData = newLen-4;
htons_newLen = htons(newLenData);
memcpy((void *)&GliHeaderData[2],(const
void*)&htons_newLen,sizeof(htons_newLen));
memcpy((void*)pNewBuf,(const
void*)&GliHeaderData,GLI_HEADER_LENGTH);
memcpy(pNewBuf+GLI_HEADER_LENGTH,pSnd,uLen);
```

Enabling the IVR Server Debug

Use the Configuration Layer to enable the IVR Server debug. In the `Application` object for the T-Server that connects to the IVR Server, on the `Options` tab, add a section named `pgf-debug`. Then add the following key-value pair to that section:

Key: `debug`

Value: `default:ALL`

Call-Scenario Examples

The remainder of this chapter presents several representative call scenarios. The description for each kind of call contains:

- A graphic showing the request-response interaction for the entire call flow.
- A step-by-step breakdown. Each step includes:
  - A sample XML message that you can use as a model or starting point for your application.
  - An explanation that points out key elements and parameters, defines certain terms, and, when necessary, explains the logic of the Genesys IVR XML protocol as it relates to the particular interaction type.

Interaction Format

As the call flows demonstrate, the conversation between the IVR, mediated through your client IVR driver application, and the Genesys IVR Server tends to follow a request-response sequence.

Note: The call flows included in this chapter are examples. The actual call flows depend on your routing strategy and may differ from the call flows given here.

For help understanding the call flow logic, see Chapter 3 on page 33, which presents all state-to-state transitions and the triggers that initiate them.

Interaction Example

The `RouteRequest` and `RouteResponse` message pair provides a good example of the request-response sequence.

Note: This example pertains to a `CallControlMode=Genesys` environment.

1. After the IVR receives an incoming call, it prompts your client application to send a `NewCall` message to Genesys IVR Server. This message contains the `CallID` that will be used throughout the entire transaction.
2. IVR Server returns the appropriate call status messages, `CallStatus (Ringing)` and `CallStatus (Established),` to the client IVR driver application which then forwards them to the IVR.
3. When the IVR has processed the call, it sends a second message, via the client IVR driver application, asking IVR Server how to route the call. IVR Server then passes the request on to the Genesys Universal Routing Server (URS). The message may include attached information, such as Customer-Entered Data (CED).

4. IVR Server sends back the `RouteResponse` message which it receives from URS. This message indicates the `RouteType`, which is the method of routing used to send the call to an agent (`Default`, `Normal`, and so on).

Further Information

- For a complete set of call flow diagrams, see Chapter 4 on page 43.
- For documentation of all messages and parameters, see Chapter 5 on page 61.
- For a complete text of the `IServer.dtd` file, see the Appendix, "The IVR Server DTD," on page 173.

A Typical Call Flow

The following call flow, shown in Figure 26, demonstrates a basic, commonly-encountered type of interaction.

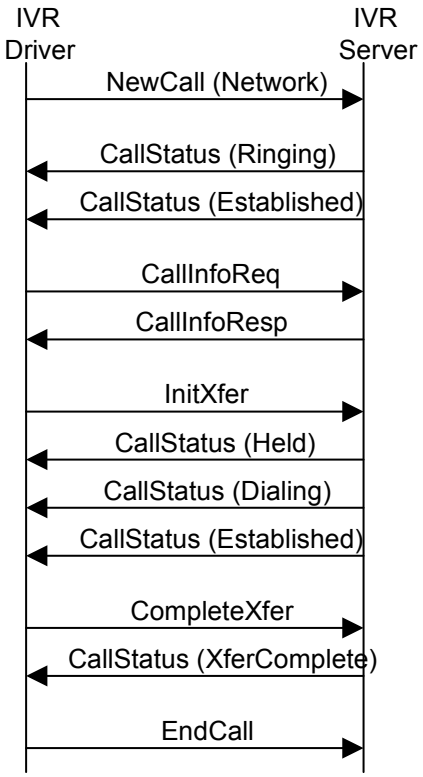

Figure 26: Typical Call Flow

The sections below include the code needed to create each step of this interaction, with explanations of key elements and parameters for each.

NewCall

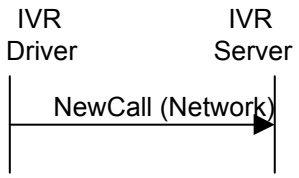

Figure 27: NewCall Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>41</CallId>
   <NewCall CallControlMode='Network' Version='3.0'>
      <CalledNum>1</CalledNum>
      <DNIS>5550700</DNIS>
      <ANI>3432232</ANI>
   </NewCall>
</GctiMsg>
```

Comments

The IVR receives the incoming call and sends the `CallID` to the client application, which uses the format shown above to transfer the information to IVR Server. The `CallID` remains the same during all phases of the interaction.

`CallControlMode` is a required parameter. The values are `Genesys` or `Network`. The `CallControlMode` parameter determines whether URS will control this call (`CallControlMode=Genesys`) or the IVR will control this call (`CallControlMode=Network`).

`Version` 3.0 indicates the current iteration of the protocol, as defined in the `IServer.dtd` file on your Genesys IVR CD-ROM.

`CalledNum`, with a value of `1`, is the IVR port configured in the Configuration Layer, under the IVR which took the call.

Note: The `CalledNum` is only a port number when dealing with In-Front or Behind mode.

This example shows only two of the optional parameters for this message, `DNIS` and `ANI`. For a complete list, see “NewCall” on page 73.

CallStatus (Ringing) and CallStatus (Established) Messages

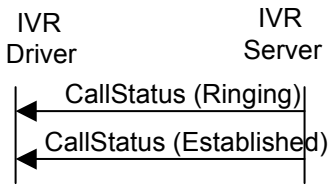

Figure 28: CallStatus (Ringing) and CallStatus (Established) Messages

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>41</CallId>
   <CallStatus Event='Ringing'/>
</GctiMsg>
<--! Next CallStatus message -->
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>41</CallId>
   <CallStatus Event='Established'/>
</GctiMsg>
```

Comments

`CallStatus` has one required parameter, `Event`, which can have one of a number of values. For a complete list, see “CallStatus” on page 84.

This example shows the `CallStatus (Ringing)` and `CallStatus (Established)` events, which follow a `NewCall` message when in Network mode. The `CallStatus (Established)` message indicates that the IVR can initiate the next step in the call flow.

CallInfoReq and CallInfoResp Messages

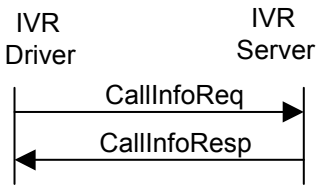

Figure 29: CallInfoReq and CallInfoResp

```
<--! The Call Info Request -->
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
```

```
<GctiMsg>
   <CallId>41</CallId>
   <CallInfoReq />
</GctiMsg>
<--! The Call Info Response -->
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>41</CallId>
   <CallInfoResp DNIS='5550700' ANI='3432232' />
</GctiMsg>
```

Comments

All `CallInfoResp` parameters are optional (implied). Whatever information is available from T-Server is supplied.

For a complete list of `CallInfoResp` parameters, see "CallInfoResp" on page 87.

InitXfer

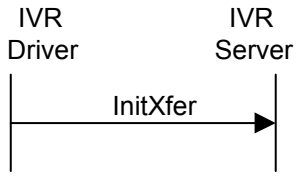

Figure 30: InitXfer Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>41</CallId>
   <InitXfer DestDN='5550800'/>
</GctiMsg>
```

Comments

You have the option to add user data and extension information to the `InitXfer` message if desired. To do so, use the `UDataEx` or `ExtnsEx` parameter.

`DestDN` is a required parameter for the `InitXfer` message.

For a full list of `InitXfer` message parameters, see "InitXfer" on page 82. See also `InitXfer` as used in "Transfer Consult Operation" on page 122.

CallStatus (Held), CallStatus (Dialing), and CallStatus (Established)

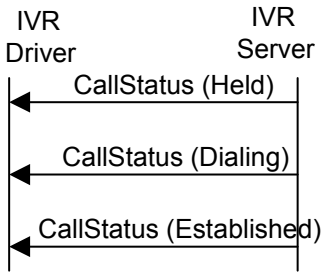

Figure 31: CallStatus (Held), CallStatus (Dialing) and CallStatus (Established) Messages

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>41</CallId>
   <CallStatus Event='Held'/>
</GctiMsg>
<--! Next CallStatus message -->
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>41</CallId>
   <CallStatus Event='Dialing'/>
</GctiMsg>
<--! Next CallStatus message -->
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>41</CallId>
   <CallStatus Event='Established'/>
</GctiMsg>
```

Comments

These messages tell the IVR, via the client IVR driver application, what is happening to the call. `CallStatus (Held)` refers to the original call, while the `CallStatus (Established)` indicate the progress in opening a new call to the transfer destination.

For a complete list of `CallStatus` event parameters, see “CallStatus” on page 84.

CompleteXfer

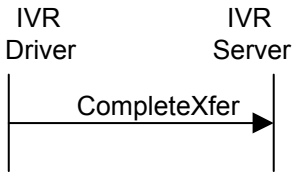

Figure 32: CompleteXfer Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>41</CallId>
   <CompleteXfer/>
</GctiMsg>
```

Comments

The IVR sends the `CompleteXfer` message after notification that the new call to the transfer destination has been established. `CompleteXfer` indicates that the original call, which has been on hold, should be connected to the transfer destination.

`CompleteXfer` has no required parameters. For a complete list of `CompleteXfer` parameters, see “CompleteXfer” on page 83.

CallStatus (XferComplete)

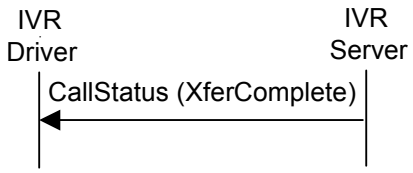

Figure 33: CallStatus (XferComplete) Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>41</CallId>
   <CallStatus Event='XferComplete'/>
</GctiMsg>
```

Comments

The Genesys IVR Server sends the `CallStatus (XferComplete)` message to the IVR, via the client IVR driver application, when the transfer has been successfully accomplished.

For a complete list of `CallStatus` event parameters, see “CallStatus” on page 84.

EndCall

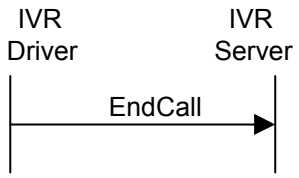

Figure 34: EndCall Message

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE GctiMsg SYSTEM "IServer.dtd">
<GctiMsg>
   <CallId>IVR_SYSy1130y261</CallId>
   <EndCall EndCause="Normal">
      <ExtnsEx>
         <Node Name="GCTIActiveRelease" Type="Str" Val="false" />
      </ExtnsEx>
   </EndCall>
</GctiMsg>
```

Comments

The `EndCause` attribute is required for the `EndCall` message. This example uses the `GCTIActiveRelease` parameter. For details about its use and a complete list of `EndCause` parameters, see “EndCall” on page 75.

Routing

Figure 35 on page 105 shows the complete call flow for a call routed by URS, contacted through the Genesys IVR Server.

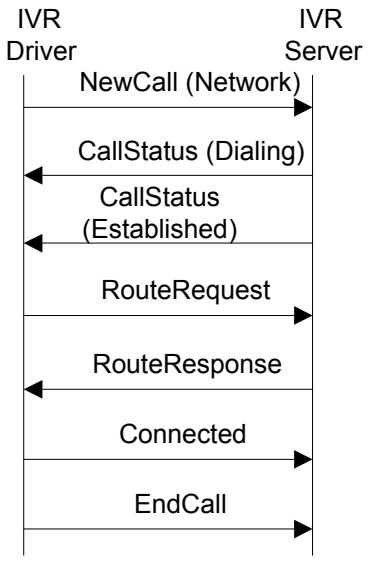

Figure 35: Routing Call Flow

The sections below include only code for steps of this interaction that have not yet been presented.

RouteRequest

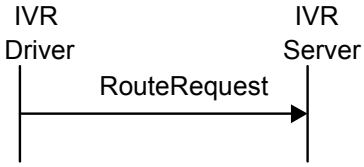

Figure 36: RouteRequest Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>12</CallId>
   <RouteRequest RouteDN='5550700'>
      <CED>1442914432</CED>
   </RouteRequest>
</GctiMsg>
```

Comments

If you are using IVR In-Front or IVR Behind, `RouteDN` is a required parameter for the `RouteRequest` message. It is optional only if you are using IVR Network mode.

This RouteRequest sample includes optional Customer-Entered Data (CED), which can be used in the routing strategy.

See "RouteRequest" on page 74 for further information about this message.

RouteResponse

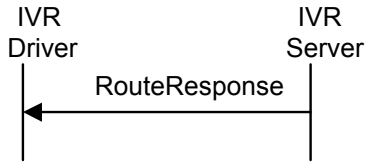

Figure 37: RouteResponse Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>12</CallId>
   <RouteResponse RouteType='Normal'>
      <Dest>5550700</Dest>
      <ExtnsEx>
         <Node Name='CUSTOMER_ID' Type='Str' Val='Test'/>
      </ExtnsEx>
   </RouteResponse>
</GctiMsg>
```

Comments

The RouteType attribute is required. See "RouteResponse" on page 74 for a complete list of Route Types.

This sample XML message shows the use of the optional ExtnsEx parameter using the Node attribute. This attribute requires Name, Type, and Val values. For more on ExtnsEx, see "ExtnsEx" on page 94.

Connected

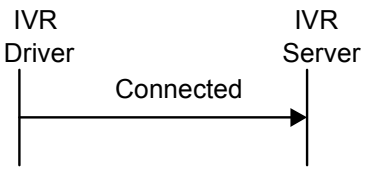

Figure 38: Connected Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
```

```
   <CallId>12</CallId>
   <Connected/>
</GctiMsg>
```

Comments

Connected has only one parameter, the optional parameter `ExtnsEx`.

Call Treatment Operation

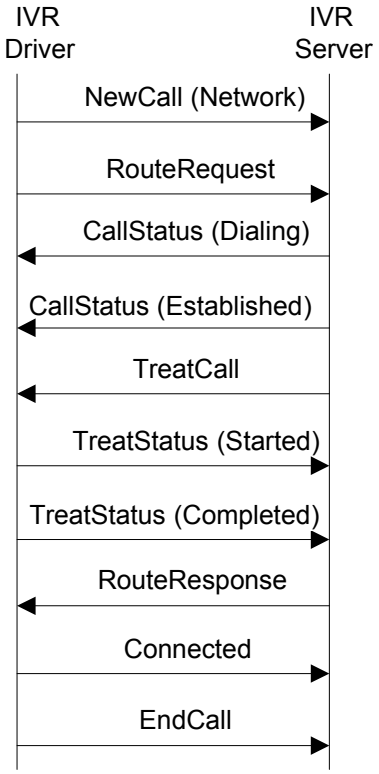

Figure 39: Call Treatment Call Flow

The call treatment call flow uses the same basic elements as the previous examples, with the addition of messages controlling a call treatment application.

TreatCall

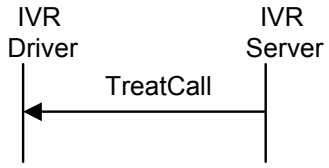

Figure 40: TreatCall Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <TreatCall Type='PlayApplication'>
      <Parameters>
         <Node Name='APP_ID' Type='Int' Val='1'/>
         <Node Name='LANGUAGE' Type='Str' Val='English(US)'/>
      </Parameters>
   </TreatCall>
</GctiMsg>
```

Comments

The `Type` parameter is required. The other parameters are optional. The content list for the optional parameters depends on the call treatment type. In this example, the treatment type is `PlayApplication` and the parameters identify the application and indicate its language.

For all `TreatCall` parameters, see “TreatCall” on page 76.

TreatStatus (Started) and TreatStatus (Completed)

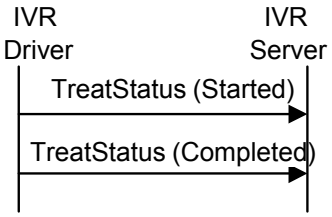

Figure 41: TreatStatus (Started) and TreatStatus (Completed) Messages

```
<--! TreatStatus (Started) Message -->
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE GctiMsg SYSTEM "IServer.dtd">
<GctiMsg>
   <CallId>3</CallId>
   <TreatStatus Status="Started"></TreatStatus>
</GctiMsg>
<--! TreatStatus (Completed) Message -->
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE GctiMsg SYSTEM "IServer.dtd">
<GctiMsg>
   <CallId>3</CallId>
   <TreatStatus Status="Completed"></TreatStatus>
</GctiMsg>
```

Comments

The required parameters for `TreatStatus` are `CallId` and `Status`. For available optional parameters, see “TreatStatus” on page 77.

MakeCall Operation

A `MakeCall` operation is one in which the IVR initiates the call, which can then be transferred after the call is answered. The call flow in Figure 42, below, shows the interaction sequence for the `MakeCall` operation, which would then be followed by whatever further operations are appropriate.

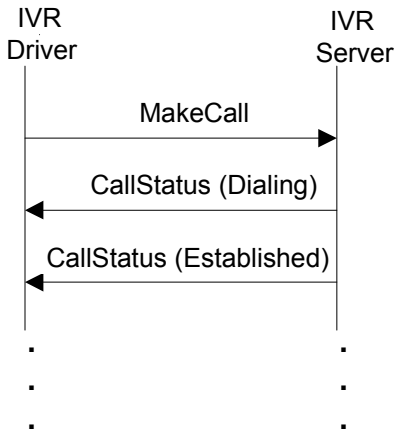

Figure 42: Call Flow for the MakeCall Operation

MakeCall

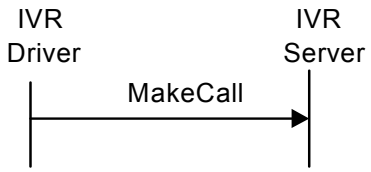

Figure 43: MakeCall Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>41</CallId>
   <MakeCall OrigNum='5550700' DestNum='3432232'/>
</GctiMsg>
```

Comments

Both `OrigNum` and `DestNum` are required parameters. `MakeCall` has no other parameters.

CallStatus (Dialing)

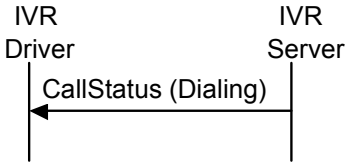

Figure 44: CallStatus (Dialing) Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>41</CallId>
   <CallStatus Event='Dialing'/>
</GctiMsg>
```

Comments

`CallStatus` has one required parameter, `Event`, which can have one of a number of values. For a complete list, see “CallStatus” on page 84.

CallStatus (Established)

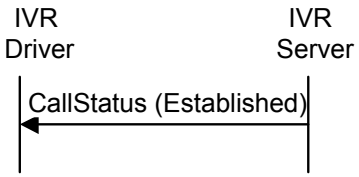

Figure 45: CallStatus (Established) Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>41</CallId>
   <CallStatus Event='Established'/>
</GctiMsg>
```

Comments

`Established` is also a valid value for the required `CallStatus` parameter, `Event`. For a complete list, see “CallStatus” on page 84.

One-Step Conference Operation

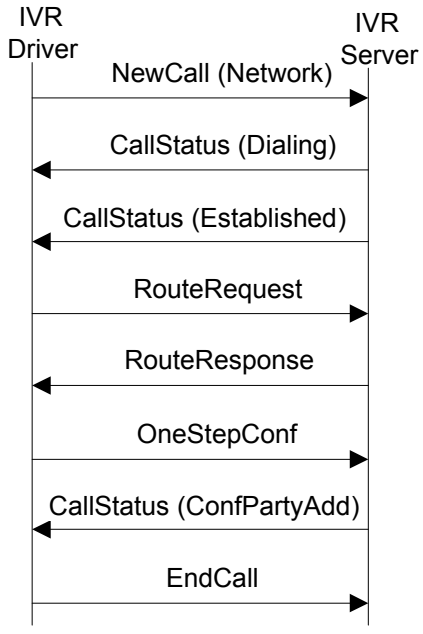

Figure 46: One-Step Conference Call Flow

One-step conferences enable immediate conferencing of active calls with destination DNs as quickly as the PBX can act. This feature can be used for chat and e-mail as well as calls.

Note: Some switches do not support this feature. If your switch does not support one-step transfers, you receive an `Unsupported Operation` error message.

OneStepConf

Figure 47: OneStepConf Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>7</CallId>
   <OneStepConf DestDN='5550800'/>
</GctiMsg>
```

Comments

`DestDN` is a required parameter. For optional parameters, see “OneStepConf” on page 82.

CallStatus (ConfPartyAdd)

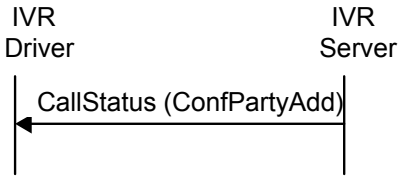

Figure 48: CallStatus (ConfPartyAdd) Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>7</CallId>
   <CallStatus Event='ConfPartyAdd'/>
</GctiMsg>
```

Comments

`ConfPartyAdd` is a valid value for the `CallStatus` message's required `Event` parameter.

Conference Consult Operation

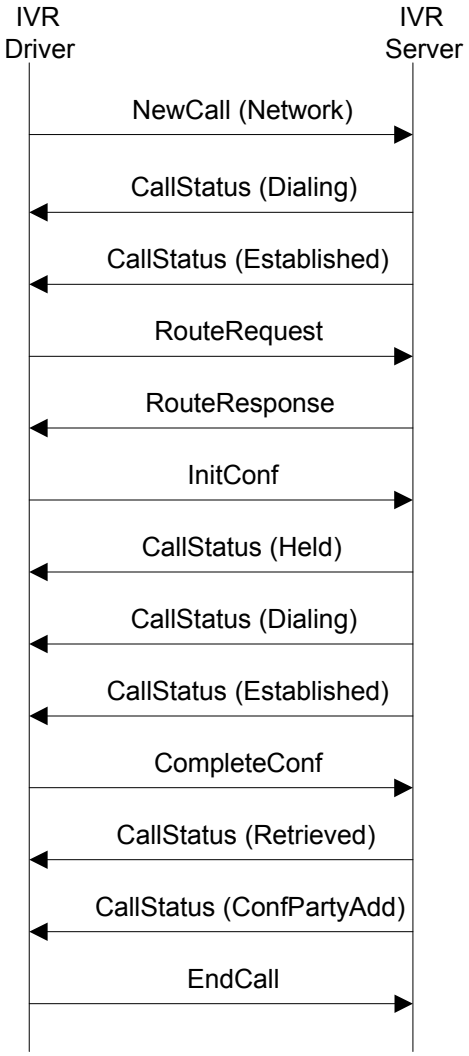

Figure 49: Conference Consult Call Flow

The Conference Consult interaction sequence enables more complex conferencing scenarios than one-step conferencing. Instead of the two calls joining as soon as the PBX can accomplish it, the initial call is placed on hold, and a second call is opened to the party that should be conferenced in. Only after the second call is established is it conferenced with the first call.

InitConf

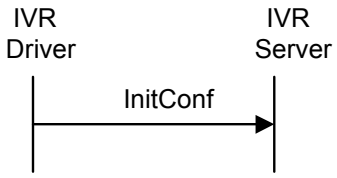

Figure 50: InitConf Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <InitConf DestDN='5550800'/>
</GctiMsg>
```

Comments

`DestDN` is a required parameter for the `InitConf` message. For optional parameters, see “InitConf” on page 83.

CallStatus (Held)

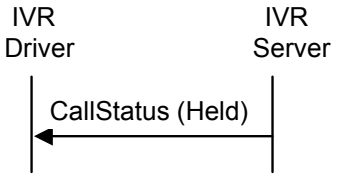

Figure 51: CallStatus (Held) Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <CallStatus Event='Held'/>
</GctiMsg>
```

Comments

`Held` is a valid value for the `CallStatus` message’s required `Event` parameter. In this case it indicates that the original call has been placed on hold in preparation for opening a new call to the party to be added to the conference.

CallStatus (Dialing)

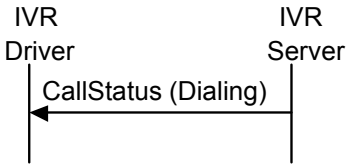

Figure 52: CallStatus (Dialing) Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <CallStatus Event='Dialing'/>
</GctiMsg>
```

Comments

This `CallStatus` message indicates that a new call has been initiated to the party to be added to the conference.

CallStatus (Established)

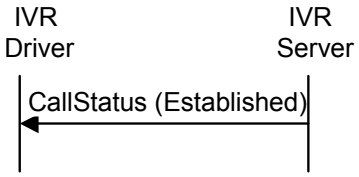

Figure 53: CallStatus (Established) Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <CallStatus Event='Established'/>
</GctiMsg>
```

Comments

This message indicates to the IVR driver and IVR that the additional call has been established and signals a ready state for completing the conference.

If at this point the call status is `Busy`, the IVR driver should send a `RetrieveCall` message. `RetrieveCall` signals that the busy call should be dropped and the original call, which was put on hold, reconnected.

Note: `RetrieveCall` is used only to reconnect an original call that was placed on hold after an `Initiate Conference` or `Initiate Transfer` message. `RetrieveCall`, if used, must occur before the `Complete Conference` or `Complete Transfer` message.

CompleteConf

Figure 54: CompleteConf Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <CompleteConf/>
</GctiMsg>
```

Comments

`CompleteConf`, which signals to IVR Server that the calls to be conferenced should be joined, does not require any parameters. If desired, you can use the optional `ExtnsEx` parameter.

CallStatus (Retrieved)

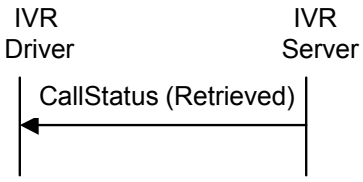

Figure 55: CallStatus (Retrieved) Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <CallStatus Event='Retrieved'/>
</GctiMsg>
```

Comments

`CallStatus (Retrieved)` indicates that the original call, which was placed on hold, has been activated and connected with the newly established call.

Note: `CallStatus (Retrieved)` and `RetrieveCall` do not have the same meaning or functions. See the note regarding `RetrieveCall` on page 116 for clarification on the difference between the two messages.

CallStatus (ConfPartyAdd)

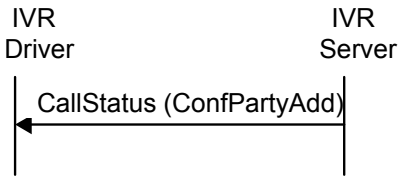

Figure 56: CallStatus (ConfPartyAdd)

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <CallStatus Event='ConfPartyAdd'/>
</GctiMsg>
```

Comments

`ConfPartyAdd` is a valid value for the `CallStatus` message's required `Event` parameter. This status message indicates that the original call has successfully been reactivated and joined in a conference with the new call.

Transfer to Remote Site Operation

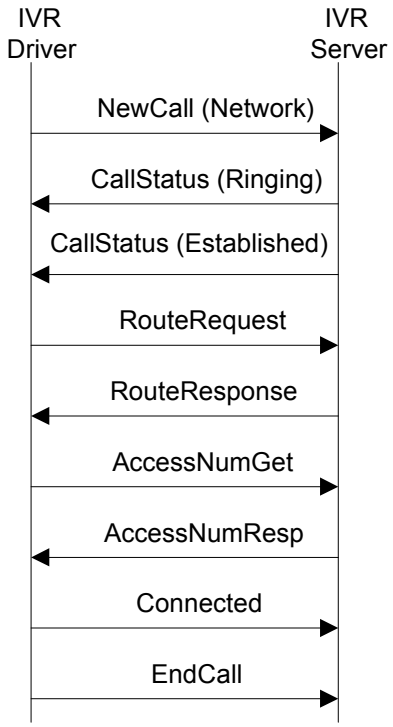

Figure 57: Transfer to Remote Site Call Flow

AccessNumGet

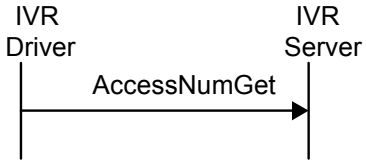

Figure 58: AccessNumGet Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>17</CallId>
   <AccessNumGet DestDN='5550700' Location='Switch_X'
    XRouteType='Default'/>
</GctiMsg>
```

Comments

The `DestDN` and `Location` parameters are required. The `XRouteType` parameter is optional.

The value for the `Location` parameter is the name given to the switch when that `Switch` object is set up in Configuration Manager. It is used by an external router.

Additional optional parameters are given in “AccessNumGet” on page 79.

AccessNumResp

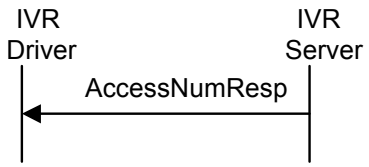

Figure 59: AccessNumResp Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>17</CallId>
   <AccessNumResp Action='Get' Result='Success' AccessNum='2200'/>
</GctiMsg>
```

Comments

The parameters `Action` and `Result` are required for the `AccessNumResp` message. The `AccessNum` parameter is optional. Additional optional parameters are given in “AccessNumResp” on page 81.

One-Step Transfer Operation

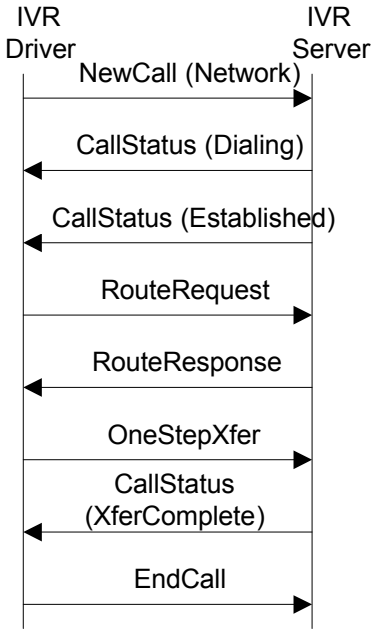

Figure 60: One-Step Transfer Call Flow

One-step transfers enable simple, immediate call transfers. They are most often used during power or predictive dialing when transfer speed is critical.

Not all switches support one-step transfer. If you receive an message indicating that this feature is not supported, use `Consult Transfer` instead.

OneStepXfer

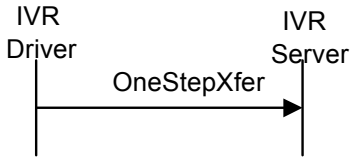

Figure 61: OneStepXfer Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>7</CallId>
   <OneStepXfer DestDN='5550800'/>
</GctiMsg>
```

Comments

`DestDN` is a required parameter for `OneStepXfer`. `Location`, `UDataEx`, and `ExtnsEx` are optional parameters for this message. See “OneStepXfer” on page 81 for details.

CallStatus (XferComplete)

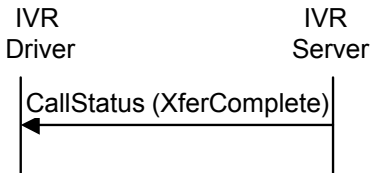

Figure 62: CallStatus (XferComplete)

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>7</CallId>
   <CallStatus Event='XferComplete'/>
</GctiMsg>
```

Comments

The `CallStatus (XferComplete)` indicates that the transfer has been successfully accomplished. For a complete list of `CallStatus` messages, see “CallStatus” on page 84.

Transfer Consult Operation

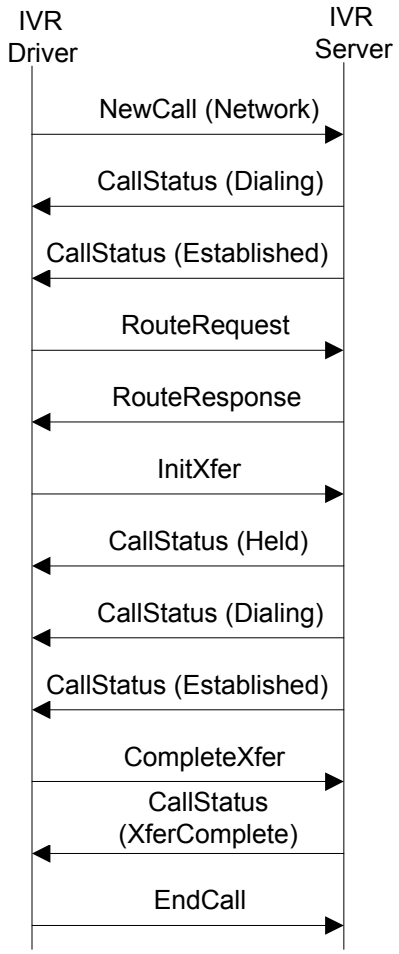

Figure 63: XferConsult Call Flow

In contrast with one-step transfer, a consultation transfer requires additional steps, but has the benefit of avoiding transfers to busy or otherwise unavailable destinations.

`Route Request` and `Route Response` are shown below to provide an example of the information that is carried into the transfer.

RouteRequest

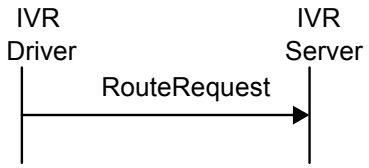

Figure 64: RouteRequest Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <RouteRequest RouteDN='8000'>
      <CED>1442914432</CED>
   </RouteRequest>
</GctiMsg>
```

Comments

If you are using IVR In-Front or IVR Behind, `RouteDN` is a required parameter for the `RouteRequest` message. It is optional only if you are using IVR Network mode.

This `RouteRequest` sample includes optional Customer-Entered Data (CED) that can be used in the routing strategy.

See “RouteRequest” on page 74 for further information about this message.

RouteResponse

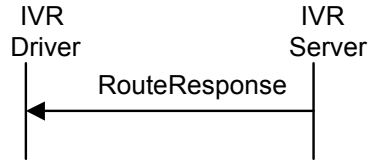

Figure 65: RouteResponse Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <RouteResponse RouteType='Normal'>
      <Dest>5550700</Dest>
   </RouteResponse>
</GctiMsg>
```

Comments

The `RouteType` attribute is required. See “RouteResponse” on page 74 for a complete list of route types. If the `RouteType` is `Default`, the message will include the default route number.

InitXfer

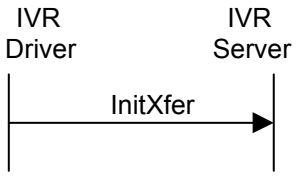

Figure 66: InitXfer Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <InitXfer DestDN='5550800'/>
</GctiMsg>
```

Comments

You have the option to add user data and extension identification to the `InitXfer` message if desired. To do so, use the `UDataEx` or `ExtnsEx` parameter.

`DestDN` is a required parameter for the `InitXfer` message.

For a full list of `InitXfer` message parameters, see “InitXfer” on page 82. See also `InitXfer` as used in “A Typical Call Flow” on page 98.

CallStatus (Held)

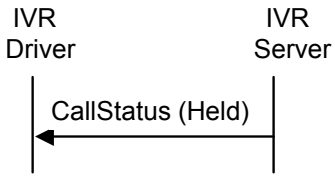

Figure 67: CallStatus (Held) Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <CallStatus Event='Held'/>
</GctiMsg>
```

Comments

`Held` is a valid value for the `CallStatus` message's required `Event` parameter. In this case it indicates that the original call has been placed on hold in preparation for opening a new call to the part to be added to the conference.

CallStatus (Dialing)

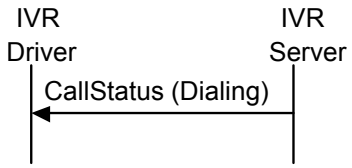

Figure 68: CallStatus (Dialing) Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <CallStatus Event='Dialing'/>
</GctiMsg>
```

Comments

This `CallStatus` message indicates that a new call has been initiated to the party that the original caller will be transferred to.

CallStatus (Established)

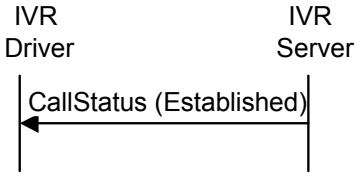

Figure 69: CallStatus (Established) Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <CallStatus Event='Established'/>
</GctiMsg>
```

Comments

This message indicates to the IVR driver and IVR that the additional call has been established and signals a ready state for completing the transfer.

If at this point the call status is `Busy`, the IVR driver should send a `RetrieveCall` message. `RetrieveCall` signals that the busy call should be dropped and the original call, which was put on hold, reconnected.

Note: `RetrieveCall` is used only to reconnect an original call that was placed on hold after an `InitiateConference` or `InitXfer` message. `RetrieveCall`, if used, must occur before the Complete Conference or Complete Transfer message.

CompleteXfer

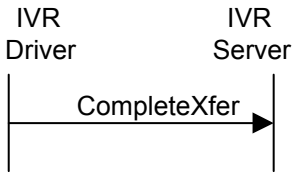

Figure 70: CompleteXfer Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <CompleteXfer/>
</GctiMsg>
```

Comments

The IVR sends the `CompleteXfer` message after notification that the new call to the transfer destination has been established. `CompleteXfer` indicates that the original call, which has been on hold, should be connected to the transfer destination.

`CompleteXfer` has no required parameters. For a complete list of `CompleteXfer` parameters, see “CompleteXfer” on page 83.

CallStatus (XferComplete)

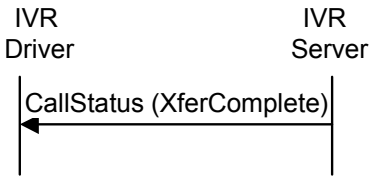

Figure 71: CallStatus (XferComplete) Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>3</CallId>
   <CallStatus Event='XferComplete'/>
</GctiMsg>
```

Comments

The `CallStatus (XferComplete)` indicates that the transfer has been successfully accomplished. For a complete list of `CallStatus` messages, see "CallStatus" on page 84.

Agent Login Interface

The following sections define an interface model relating to the agent protocol messages described in Chapter 5, "IVR XML Protocol Messages and Parameters," on page 61. This basic model that explains both server and client responsibilities.

Server Side Model

The server in this implementation behaves primarily as a proxy. The server does provide translation from port numbers to associated DN and T-Server pairs. However, the server does not maintain any state information on behalf of either the IVR or the T-Server.

XML messages arriving from the client receive basic validity checks. If those checks pass, the messages is translated to its T-Library counterpart and submitted to the relevant T-Server. Supplied reference identifiers are stored for use in reply messages from the T-Server. Messages received from the T-Server are translated to XML and sent to any interested clients.

Client Side Model

The client software has two primary responsibilities: respecting remote server state and ensuring desired agent login state. These goals have some overlap. For instance, when the remote server is either unavailable or disconnected, all agent control messages will fail. Additionally, you must be aware of the remote server name in order to correctly process server information events.

You should expect that, upon registering for server monitoring, `MonitorInfo` messages can arrive at any time. These messages can occur whenever a significant event occurs on the remote T-Server. Such significant events may not be in response to an XML request.

Login

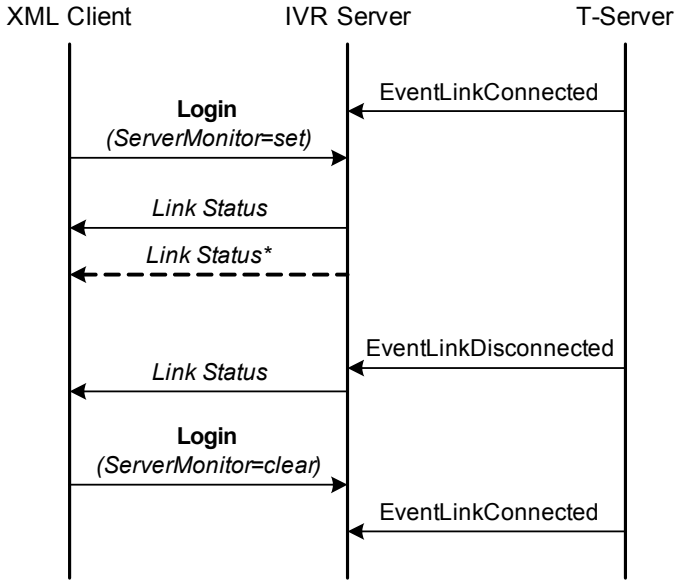

Figure 72: Login Message

Login Example Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>1</CallId>
   <LoginReq Version='4.0' ClientName='IVR_1' ServerMonitor='set' />
</GctiMsg>
```

Link Status Example Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
```

```
   <CallId>1</CallId>
   <MonitorInfo>
      <Server Name='G3_TServer_1' Status='Connected' />
   </MonitorInfo>
</GctiMsg>
```

Comments

Figure 72 on page 128 details the behavior of the server monitoring features of `Login`. When a login request that specifies monitoring is received, the current server status is sent to the requestor. As a single IVR may be interested in more than one T-Server, multiple initial server statuses may be reported. This is indicated by an asterisk(*). The figure also demonstrates the deregistration process, for clients who are no longer interested in status updates.

Port Status

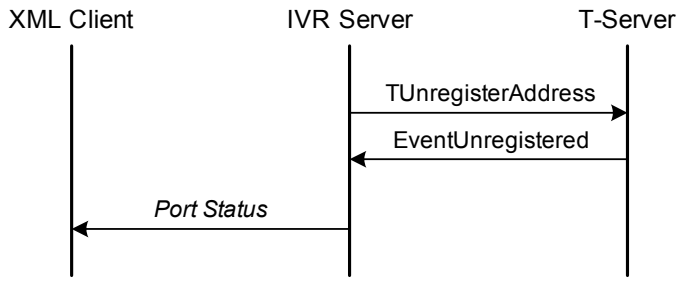

Figure 73: Port Status Message

Port Status Example Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>1</CallId>
   <MonitorInfo>
      <Server PortNum='01' Status='Unavailable' />
   </MonitorInfo>
</GctiMsg>
```

Comments

Figure 73 above is an example of generation of the port status update. This can currently only occur when the configuration is change such that a previously registered DN is no longer part of the configuration. This occurrence should be exceptionally rare.

Agent State Query

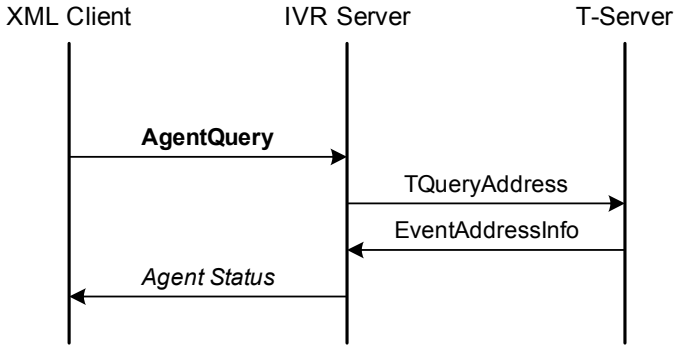

Figure 74: Agent Query Message

Agent Query Example Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>1</CallId>
   <AgentQuery ReqId='705' PortNum='01' />
</GctiMsg>
```

Agent Status Example Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>1</CallId>
   <MonitorInfo ReqId='705'>
      <Agent PortNum='01' Status='Unknown'/>
   </MonitorInfo>
</GctiMsg>
```

Comment

Figure 74 above shows an agent state query. This should always be done before attempting any other control operations. Many T-Servers will produce errors when attempting to set the agent into a state it is already in. For example, logging in a logged in agent will often produce `EventError`.

Agent Control

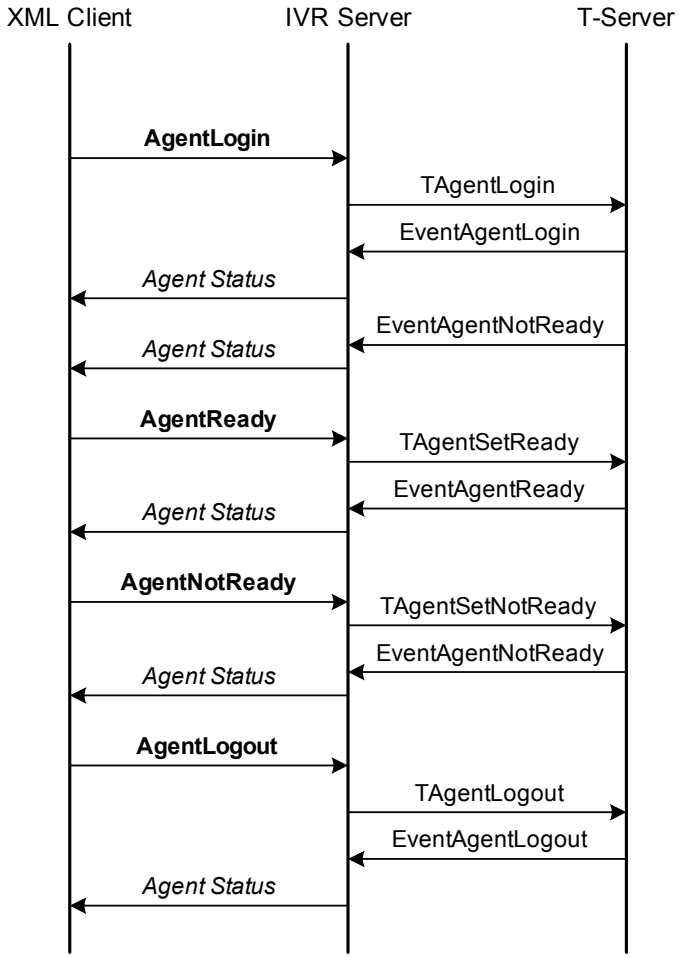

Figure 75: Agent Control Message

AgentLogin Example Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>1</CallId>
   <AgentLogin ReqId='705'  PortNum='01'
               Queue='8000' AgentId='553'
               Password='JoeyTunaFish' />
</GctiMsg>
```

AgentLogout Example Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>1</CallId>
```

```
   <AgentLogout ReqId='705'  PortNum='01' Queue='8000' />
</GctiMsg>
```

AgentReady Example Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>1</CallId>
   <AgentReady ReqId='705'  PortNum='01'
               Queue='8000' WorkMode='ManualIn' />
</GctiMsg>
```

AgentNotReady Example Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>1</CallId>
   <AgentNotReady ReqId='705'  PortNum='01'
                  Queue='8000' WorkMode='ManualIn' />
</GctiMsg>
```

Comment

Figure 75 above shows the remaining agent control messages. When an `AgentLogin` request is sent, two status messages will always follow. The first indicates the success of the login, the second the readiness state.

Error Case

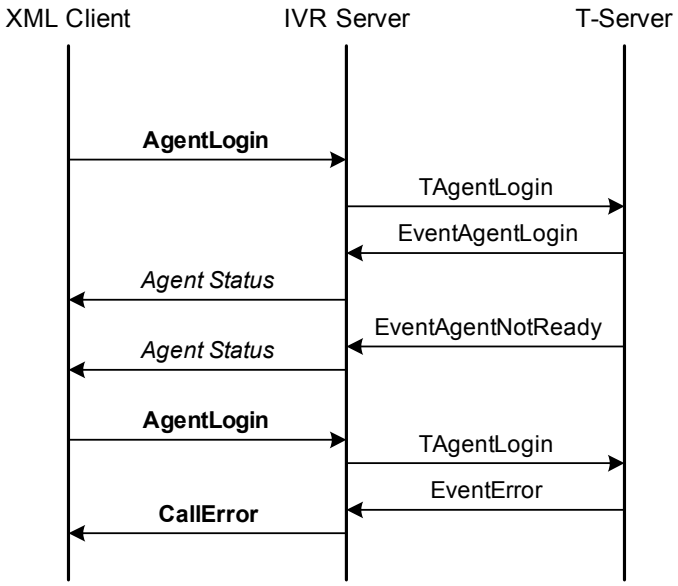

Figure 76: Error Case Message

CallError Example Message

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>1</CallId>
   <CallError FailedReq='AgentControl'
              TlibErrCode='50'
              ReqId='705' />
</GctiMsg>
```

Comment

Figure 76 is a possible error case. In this example, the T-Server generates an error because the agent state is already in the requested state. Certain T-Servers will behave in this way.

Outbound Dialing

Registration

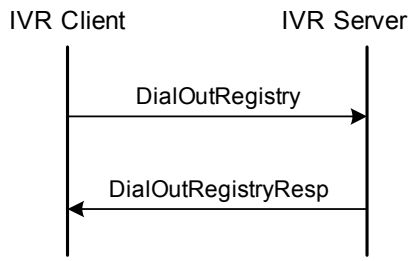

Figure 77: Registration

Dial Out Registry Example

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>1</CallId>
   <DialOutRegistry Command='Add' DN='1000'/>
</GctiMsg>
```

Dial Out Registry Resp Example

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <CallId>1</CallId>
   <DialOutRegistryResp Result='Success'/>
</GctiMsg>
```

Comment

All outbound registration requests result in a similar response. There is no specific message related to error cases. See "DialOutRegistryResp" on page 92 for details on error indications.

Request Timeout

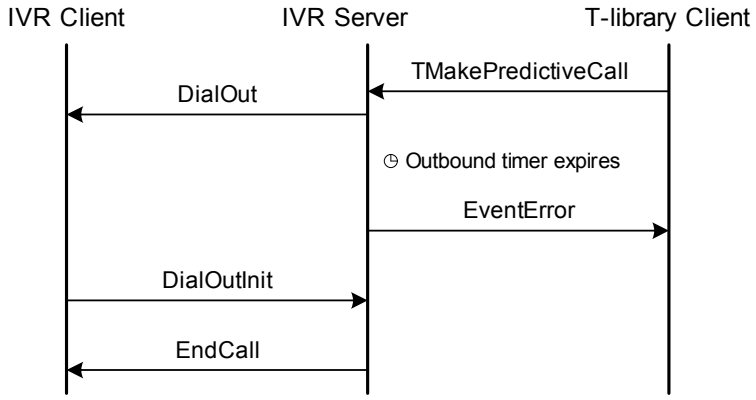

Figure 78: Dial Out Request Timeout

Dial Out Example

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
  <DialOut RefID='19' OrigNum='1000' DestNum='8435551023'/>
</GctiMsg>
```

Comment

Figure 78 above represents cases in which the IVR does not respond to a `DialOut` request. The timer duration is derived from the `AttributeTimeout` value supplied in the `TMakePredictive` call request. A late arriving `DialOutInit` is shown for informational purposes.

Dialer Error

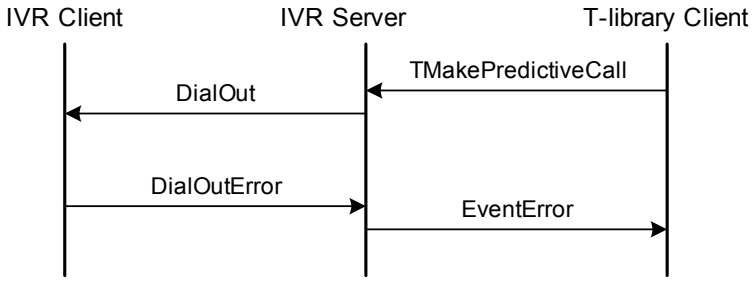

Figure 79: Dial Out Error

Dial Out Error Example

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <DialOutError RefID='19' Error='NoTrunks'/>
</GcitMsg>
```

Comment

Figure 79 above deals with `DialOut` rejection by the IVR. For cases in which the client cannot or will not make an outbound call, an error is returned.

Connection Failure

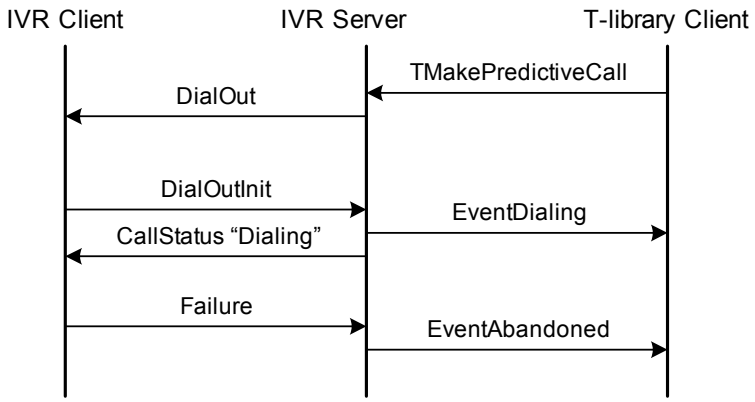

Figure 80: Connection Failure

Dial Out Init Example

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE GctiMsg SYSTEM 'IServer.dtd'>
<GctiMsg>
   <DialOutInit RefID='19'>
      <CalledNum>07</CalledNum>
   </DialOutInit>
</GctiMsg>
```

Comment

Unlike the error case in Figure 79, in Figure 80 above the failure occurs after the outbound call has been dialed (and thus `DialOutInit` sent). At this point the call is over.

Successful Call Flow

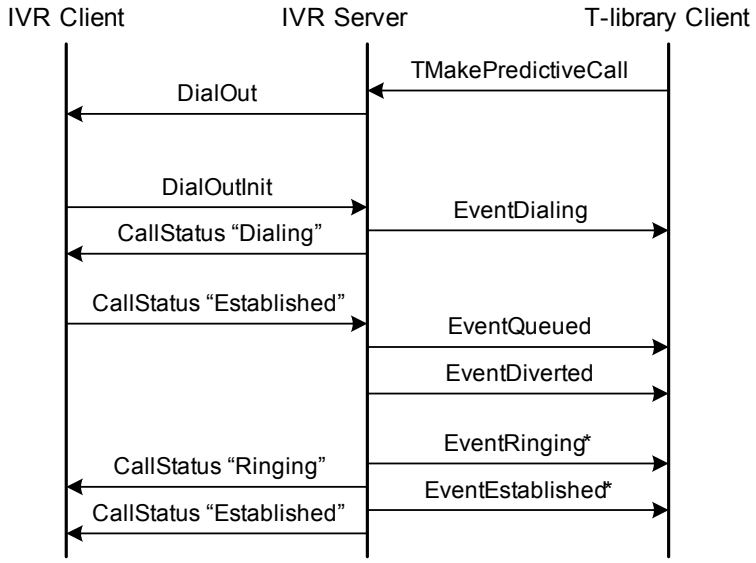

Figure 81: Successful In-Front call flow

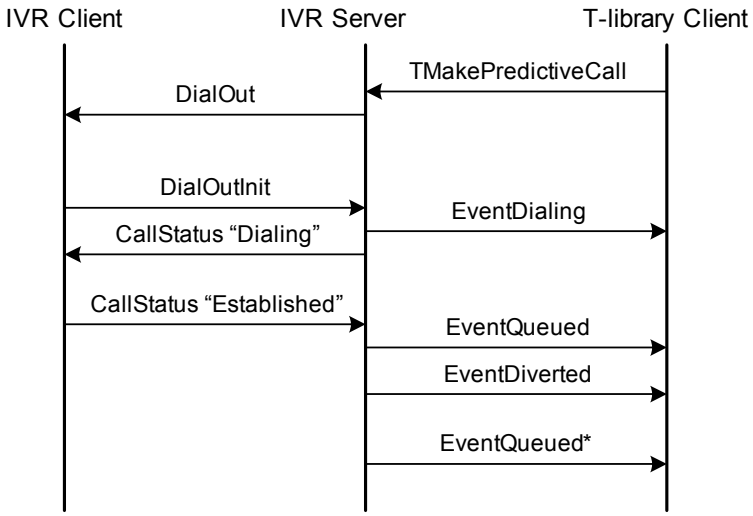

Figure 82: Successful network call flow

Comment

Figures 81 and 82 above show the case of a successful outbound call; the call is left in a state identical to that which occurs after `NewCall`. As such, routing and other call functions can proceed normally. The events marked with an asterisk (*) indicate that `AttributeThisDN` will be potentially different than in previous T-Library events. This value will be determined based on the `DialOutInit` information.

Genesys

Part

3 IVR Server Network Mode

Part Three of this *IVR SDK 8.5 XML Developer's Guide* provides details about the state model; information about specific, selected paths through the call control state machine; and protocol messages used when developing an IVR Server client that will be used in a Network mode deployment.

The information in Part Three is divided among the following chapters:

- Chapter 7, "IVR Network State Machine Diagrams," on page 139, contains state machine diagrams from the viewpoint of an IVR Server deployed in Network mode.
- Chapter 8, "Network Call Flow Diagrams," on page 147, provides call flow diagrams for many common scenarios relevant to Network mode. It is intended as a reference.
- Chapter 9, "IVR XML Protocol Messages and Parameters," on page 157, contains tables showing the parameters for each message, the message direction, and whether the parameters are required or optional when used in Network mode.

Genesys

Chapter

7 IVR Network State Machine Diagrams

This chapter details the state model to use when developing an IVR Server client that will be used in a Network mode deployment. This chapter contains these sections:

Note: The messages in the diagrams are designed to represent typical messages that your IVR sends to your IVR driver client application. The messages might differ somewhat from those given below, depending on the IVR hardware and software your enterprise uses. Notice that the `EndCall` message can arrive at anytime from the IVR after the call has started.

Call Control

The primary state machine for interactions with IVR Server is described below and illustrated in Figure 83 on page 140. The events shown in the diagram that are in italics and that start with the by the letters `IVR` are generated by an unspecified network event. These are outside the scope of this discussion. All other events are displayed in bold, and they are named to match the corresponding IVR Server XML message, detailed in Chapter 9, "IVR XML Protocol Messages and Parameters," on page 157. The numbered transitions related to network indications have the recommended actions explained below.

In cases where the transition is caused by an IVR Server message, the cause of the message is discussed instead.

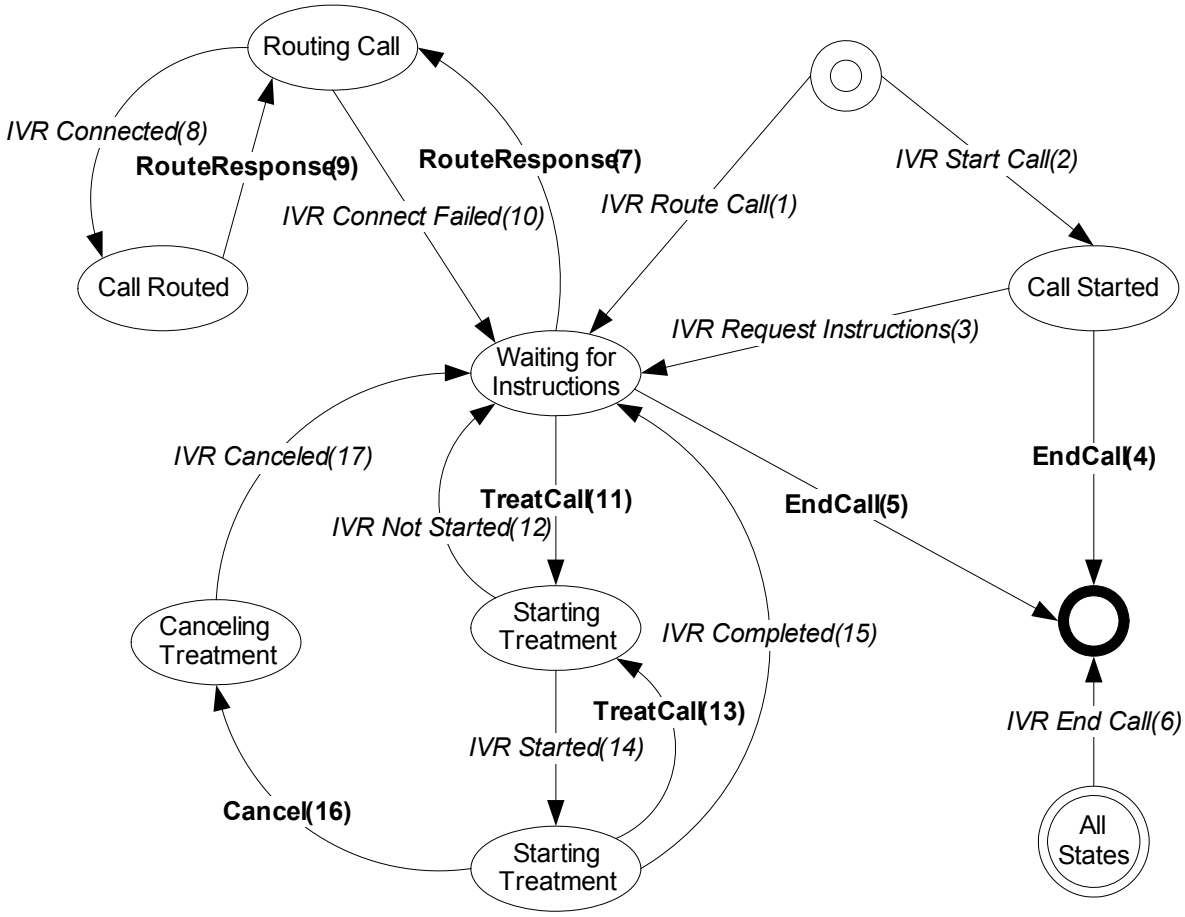

Figure 83: Call Control State Machine

This particular state machine must be active, for the following state machines to be available:

- "Call Information" on page 143
- "Logging" on page 144
- "Statistics" on page 144
- "User Data Control" on page 145

When this state machine is ended, all supplementary state machines will terminate. Outstanding requests on supplementary state machines may not be answered after the client indicates that the call is over.

Transition 1: IVR Route Call

This is one of the two forms of indicating that a call has arrived at the network platform. This form allows the client to indicate that the call has started and also that Genesys Universal Routing Server should provide routing

instructions. The client informs IVR Server of this occurrence by using the `NewCall` XML message with `CallControlMode` set to `Genesys`. This, in turn, generates an `EventRouteRequest` message.

Using this form of new call indication does not provide the same T-Library event model as using a `CallControlMode` of `Network`. The network mode indication provides an additional `EventQueued` message that is not present here.

Transition 2: IVR Start Call

The second of two forms of new call notification, this form indicates that a call has started, but that no further action is needed. The client informs IVR Server of this by issuing a `NewCall` message with `CallControlMode` set to `Network`. An `EventQueued` message is generated based upon this message.

Transition 3: IVR Request Instructions

The network platform indicates that it requires instructions from Genesys Universal Routing Server, via an `EventRouteRequest` message. A `RouteRequest` XML message is submitted to IVR Server at this point.

Transition 4: End-all

Receiving an `EndCall` message from IVR Server here will always have the same `EndCallCause`: `FeatureNotSupported`. This indicates that the `CalledNum` value specified in the `NewCall` message is not a configured route type DN. The `Switch` object associated with IVR Server must have all relevant route DNs configured. In addition, these route DNs must be registered by a T-Library client, likely Universal Routing Server.

Transition 5: EndCall

In cases where the `EndCallCause` is `FeatureNotSupported`, see the description in "Transition 4: End-all" above.

An `EndCall` message might also be received from IVR Server due to a timeout when waiting for Universal Routing Server. In the case where Universal Routing Server does not provide routing or treatment instructions within a configured period, an `EndCall` message will be sent with a cause of `Timeout`. The network platform should treat this as an instruction to handle the call using default routing.

Transition 6: IVR End Call

At any point during a call, the network platform must indicate to IVR Server that a call has ended. When this occurs, the client sends an `EndCall` XML message to IVR Server using whichever cause is appropriate.

Transition 7: RouteResponse

Universal Routing Server has indicated, via IVR Server, that the call should be routed. The routing request received from Universal Routing Server is packaged in the `RouteResponse` XML message.

Transition 8: IVR Connected

The network platform has successfully routed the call to the destination. The client then must send an XML `Connected` message to IVR Server. This message will cause an `EventRouteUsed` message to be sent to Universal Routing Server, thus ending the running strategy.

Transition 9: RouteResponse

An unsolicited `RouteResponse` message indicates that reroute has been requested. The network platform must expect this message at any time after a call is connected.

Transition 10: IVR Not Connected

The network platform is unable to route the call to the prescribed destination. The cause of the failure must be provided to IVR Server in an XML `Failure` message.

Transition 11: TreatCall

Universal Routing Server (URS) has indicated, via IVR Server, that the call should be treated by the IVR. The request received from Universal Routing Server is packaged in the `TreatCall` XML message.

Transition 12: IVR Not Started

Unable to treat the call as requested by IVR Server, the network platform indicates the treatment is not running. A `TreatStatus` message is returned to IVR Server with `Status` set to `"NotStarted"`.

Transition 13: TreatCall

Similar to “Transition 11: TreatCall” above, Universal Routing Server has indicated that the currently running treatment be stopped and a new treatment be started.

Transition 14: IVR Started

The requested treatment has been started by the client. A `TreatStatus` message must be sent to IVR Server with `Status` set to `"Started"`.

Transition 15: IVR Completed

The treatment running in the network is complete and the client needs further instructions. A `TreatStatus` message is returned to IVR Server with `Status` set to `Completed`.

Transition 16: Cancel

IVR Server has indicated that a currently running treatment should be canceled.

Transition 17: IVR Canceled

The network platform has indicated that a request to cancel a running treatment was successful. A `CancelCompleted` message is returned to IVR Server to indicate that the client is ready for further instructions.

Call Information

At any time during a call, a digest of call information may be requested. This operation cannot fail; if the specified call exists, IVR Server will return the call information in a `CallInfoResp` XML message.

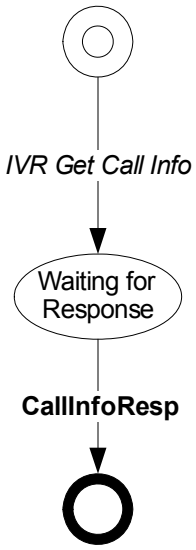

Figure 84: Call Information State Machine

Logging

The logging functionality of IVR Server does not imply a state model. No response, if within a running call, will be returned. The provided message is simply logged at the specified detail level. See “LogMsg” on page 170 for more information about message parameters.

Statistics

The act of requesting statistics primarily requires proper definition of the statistic in the IVR Server configuration. Assuming that a call has been properly started and is currently in progress, all requests will receive a `StatResp` message from the server. Outside of the scope of a call the error would be a standard `CallError` message.

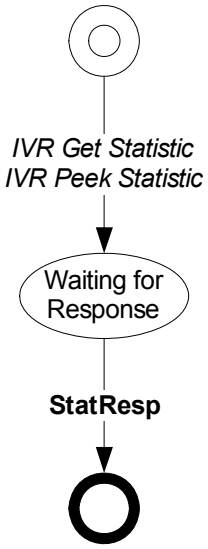

Figure 85: Statistics State Machine

Due to the simplicity of this state machine, no specific transitions need be discussed. Refer to the “PeekStatReq” on page 167 and “GetStatReq” on page 166 message definitions for information on constructing a request. Upon receiving a valid request the IVR Server will respond with a `StatResp` message. These messages should be correlated by the client based upon the `RequestId` element, which should be unique for all outstanding requests. Uniqueness of the `RequestId` is the responsibility of the network platform.

User Data Control

T-Library user data for a call can be controlled in a limited fashion. Unlike other modes of operation for IVR Server, no deletion is supported, and only one form of update is supported.

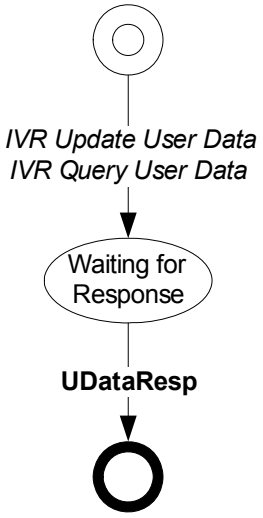

Figure 86: User Data Control State Machine

In the same fashion as the statistics state machine; requests must possess a unique `RequestId`. See "UDataGet" on page 168 and "UDataSet" on page 169 for information about request structure.

Error Responses

If at any time a message is received which is in error; a `CallError` message will be returned. The reasons for this error will be present in the message's attributes. See "CallError" on page 170 for details.

Chapter

8 Network Call Flow Diagrams

This chapter provides detailed information about specific, selected paths through the call control state machine described in the previous chapter. These paths have specific relevance to a Network mode deployment.

This chapter contains these sections:

Overview

The diagrams in this chapter detail the entire call flow, including interaction with Genesys Universal Routing Server. Since router is a required component of Network mode IVR Server operation, expect an `EndCall` (`EndCause=FeatureNotSupported`) message to be returned when Universal Routing Server is not present.

Simple Routing (Network Control)

This call flow demonstrates a simple route request/route response operation. In this particular call flow the `CallControlMode=Network`, which introduces an `EventQueued/EventDiverted` pair not present when the `CallControlMode=Genesys`.

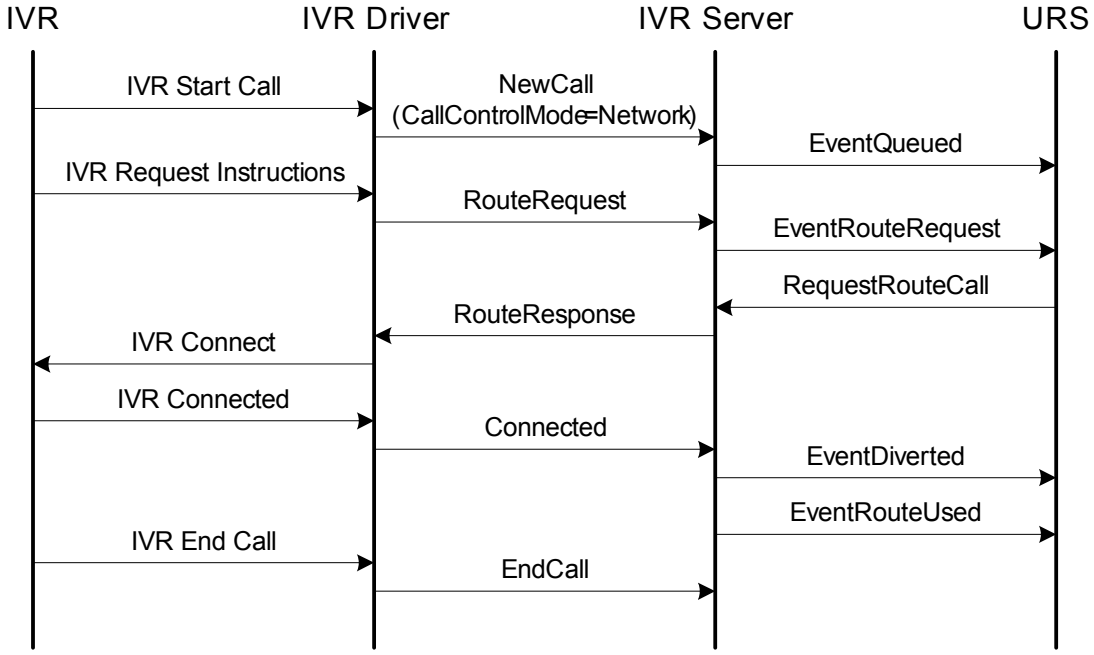

Figure 87: Simple Routing (Network) Call Flow

Simple Routing (Genesys Control)

This call flow demonstrates a simple route request/route response operation, where the `CallControlMode=Genesys`.

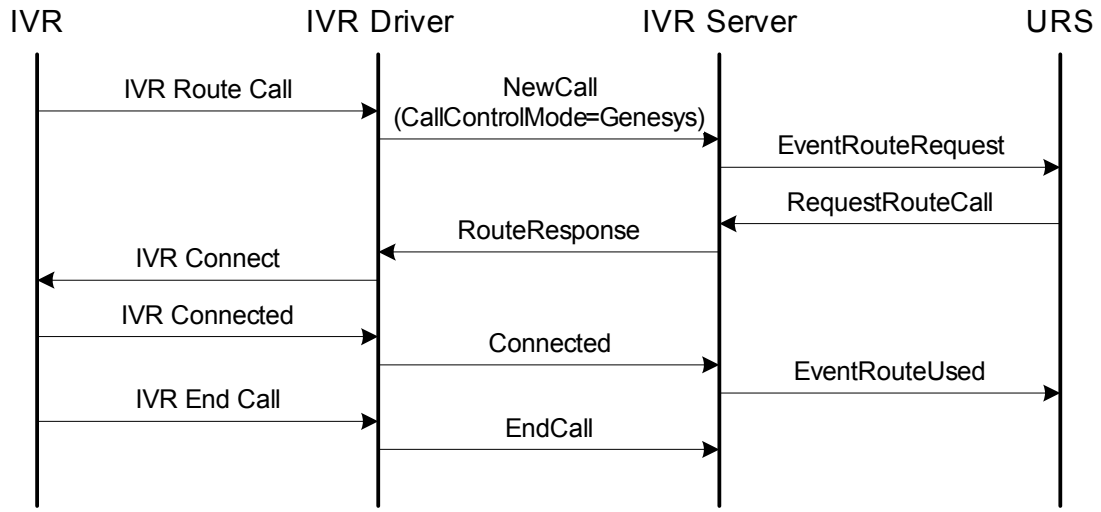

Figure 88: Simple Routing (Genesys) Call Flow

Failed Routing

In this example, two key behaviors are documented:

- The ability of the IVR driver to indicate that an error has occurred
- The behavior of `EndCall` when received prior to `Connected`

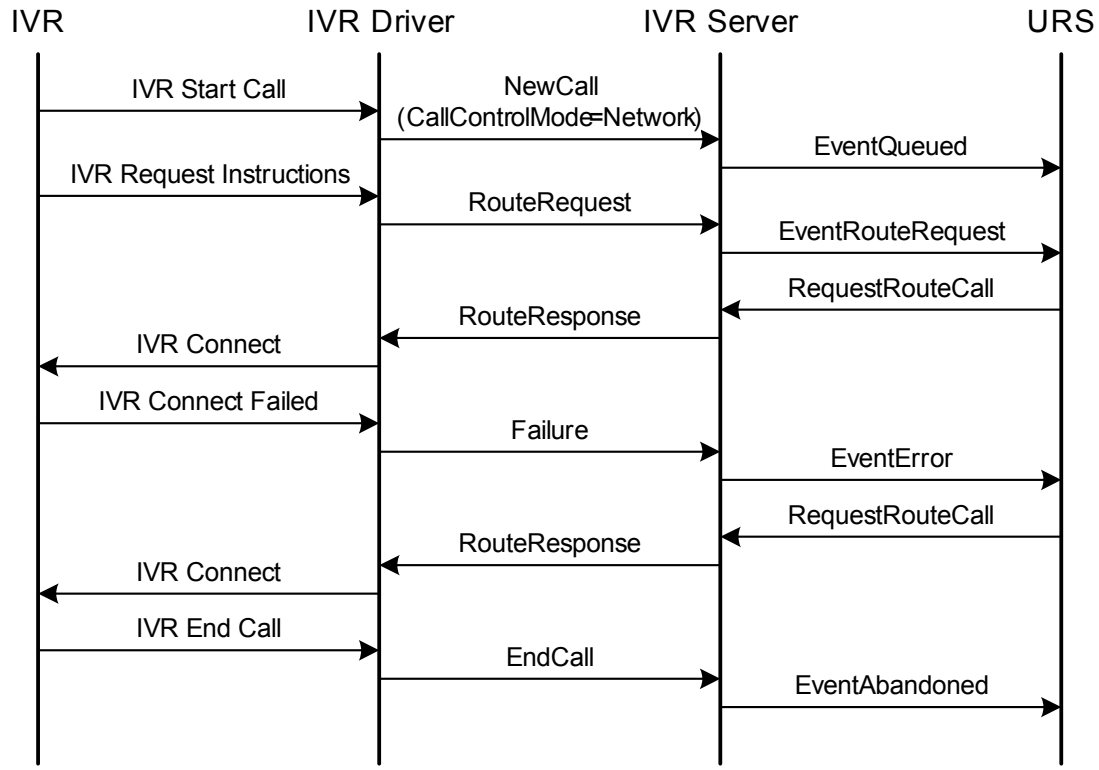

Figure 89: Failed Routing Call Flow

Routing Timeout

In cases where Universal Routing Server does not respond to a route request in a timely fashion an `EndCall` message is returned to the driver. It is important that the driver recognize that when `EndCause` is `Timeout`, this should be treated as default routing instructions. However, since IVR Server has ended its call already, `EndCall` is necessary to indicate that no further interaction attempt on the original call ID should be attempted.

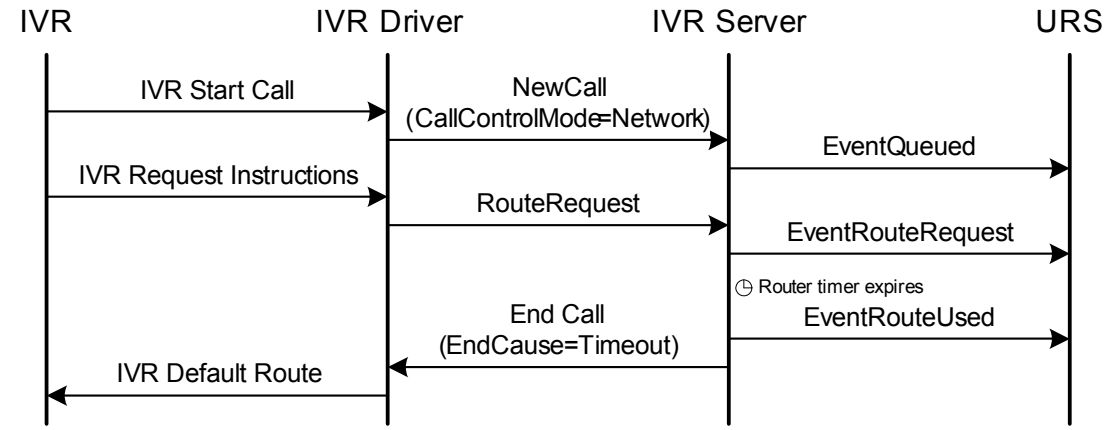

Figure 90: Routing Timeout Call Flow

Simple Treatment

Figure 91 shows a basic treatment request with routing instructions provided after treatment completion.

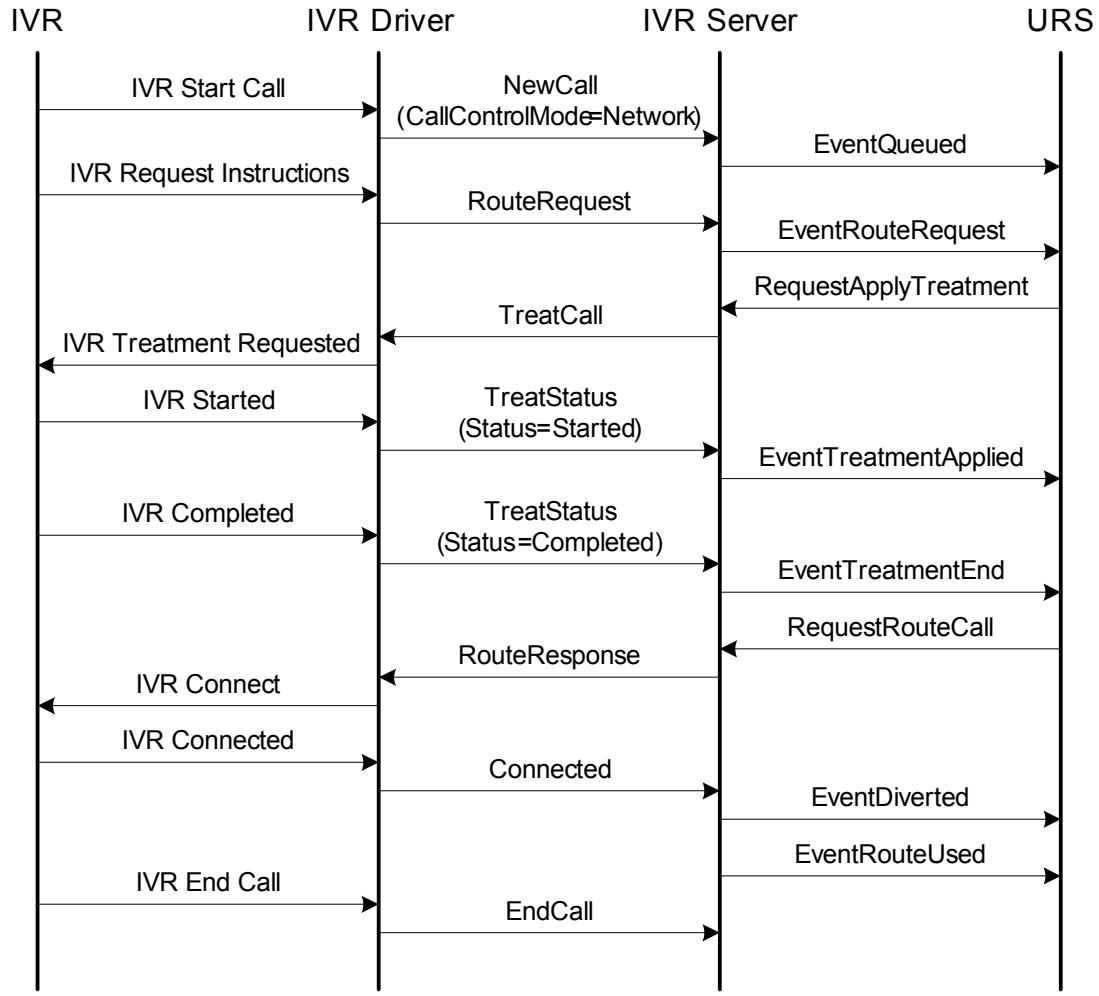

Figure 91: Simple Treatment Call Flow

Failed Treatment

Figure 92 depicts that the initial treatment request is not started.

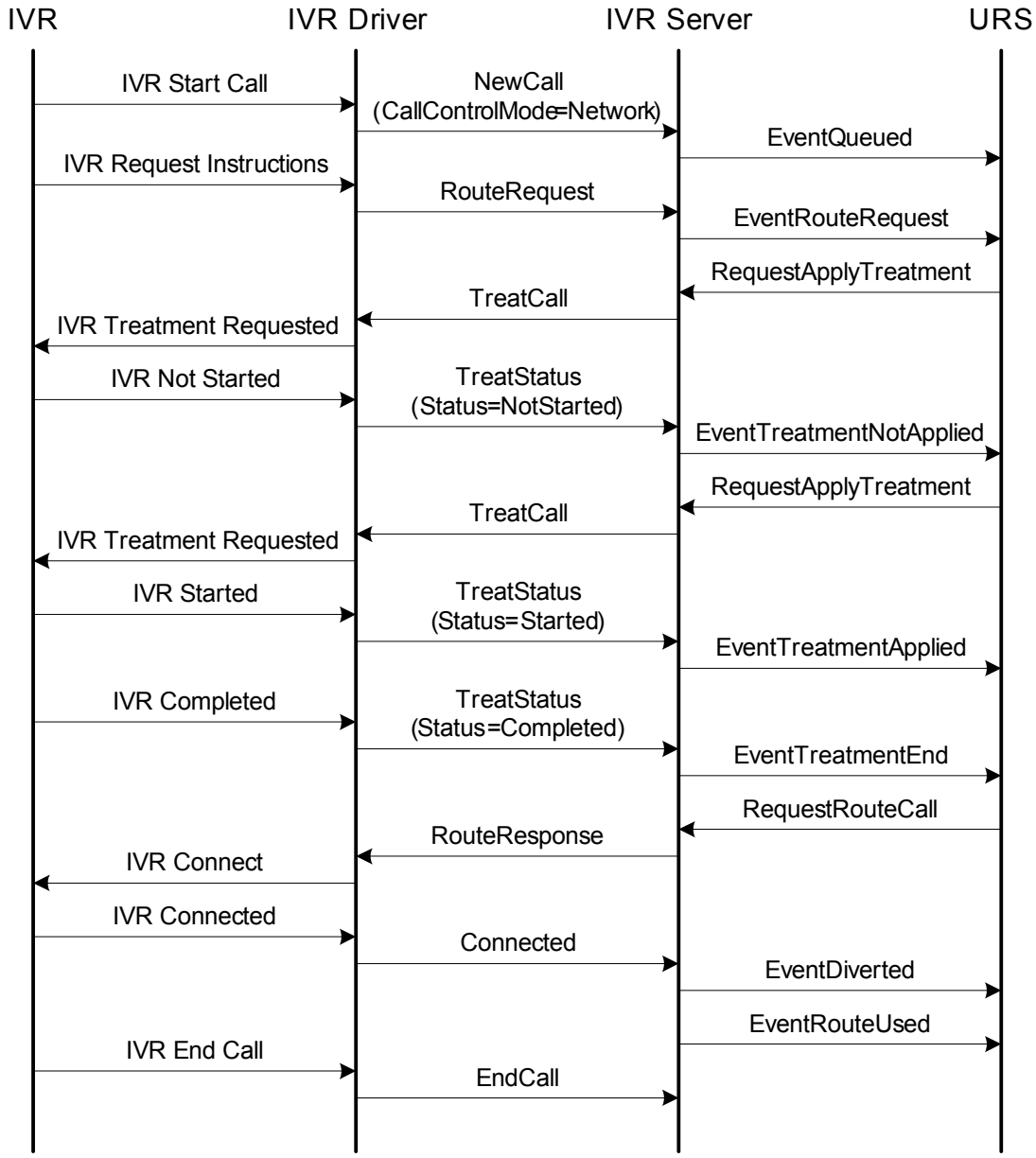

Figure 92: Failed Treatment Call Flow

Treatment Interrupted by a Routing Request

When a treatment is running on the network platform, Universal Routing Server can interrupt this treatment. When this interruption is due to available routing instructions, the treatment is actively canceled. This example is shown in Figure 93.

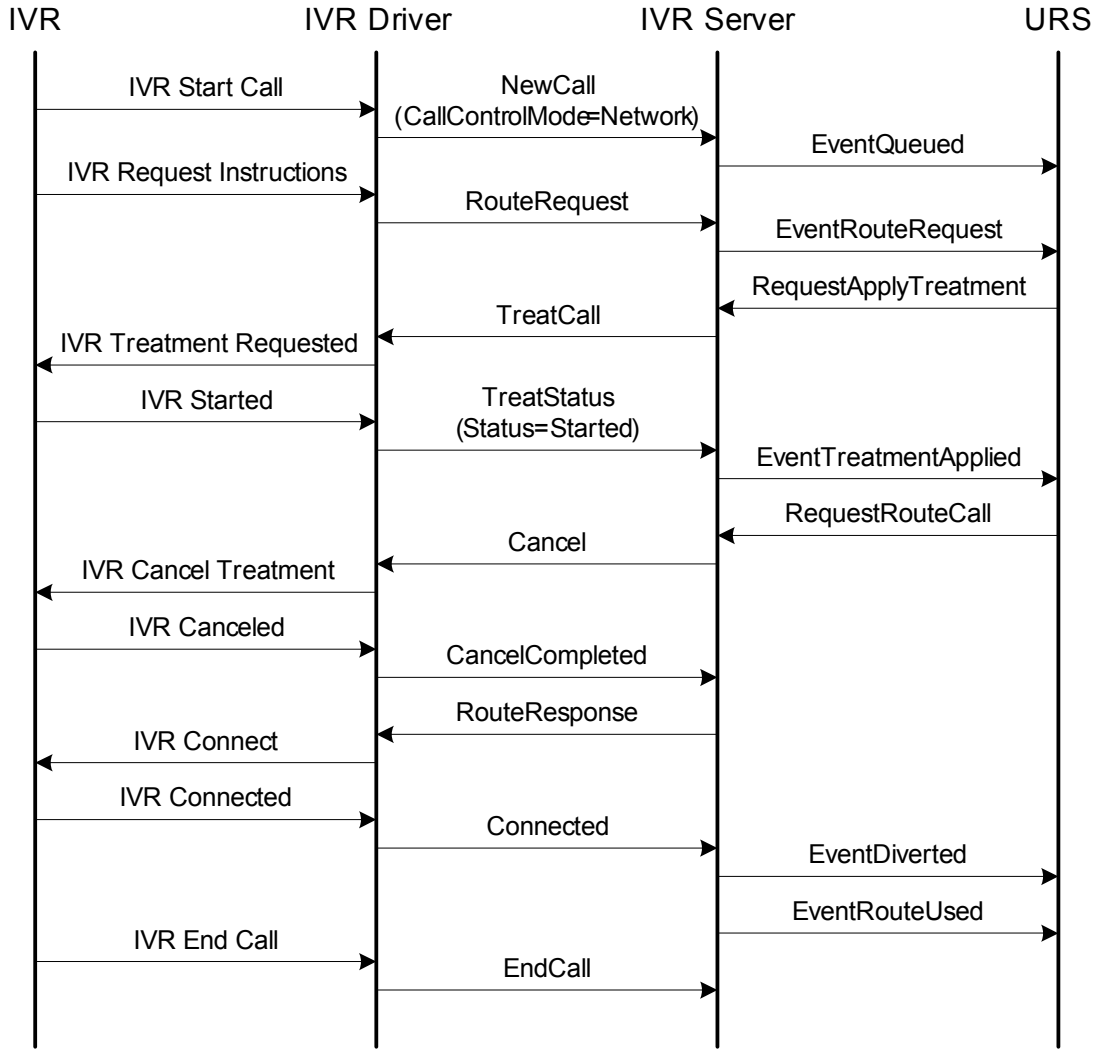

Figure 93: Treatment Interrupted by a Routing Request Call Flow

Treatment Interrupted by Another Treatment

In contrast to case above, when Universal Routing Server interrupts a treatment with another, no cancellation request is sent. When this occurs, notice that messaging related to the currently running treatment is no longer required. As such, a `TreatStatus` (`Status=Completed`) message should not be sent relating to the interrupted treatment.

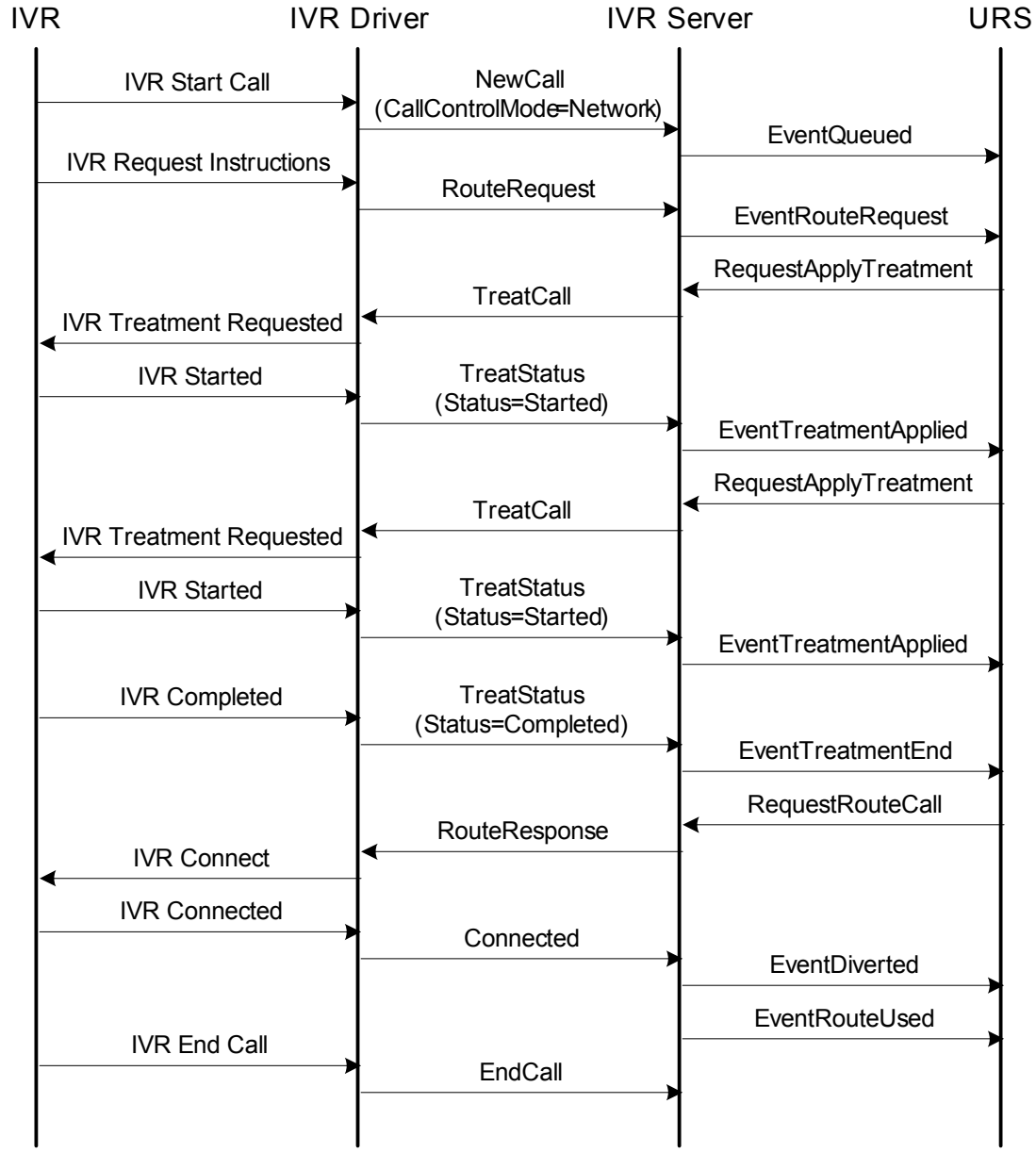

Figure 94: Treatment Interrupted by Another Treatment

Unsolicited Connect

Also referred to as network reroute, the diagram in Figure 95 shows behavior related to previously routed calls requiring a new connection.

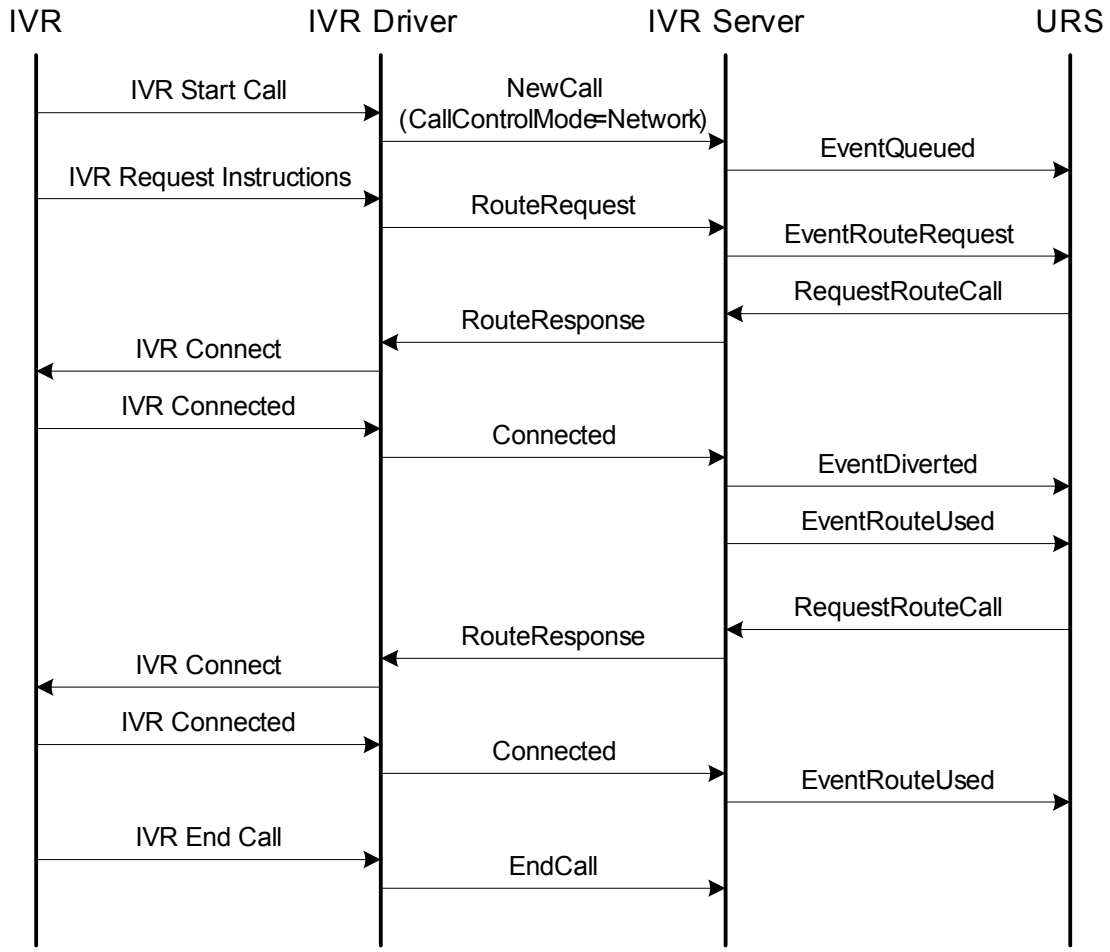

Figure 95: Unsolicited Connect Call Flow

Genesys

Chapter

9 IVR XML Protocol Messages and Parameters

This chapter presents detailed explanations of the messages and parameters used by the Genesys IVR XML protocol in a Network mode deployment situation.

This chapter contains these sections:

Overview

All messages processed by IVR Server must have a meaningful identifier. This `CallId` field uniquely identifies a particular call that conforms to the state model described in Chapter 7, "IVR Network State Machine Diagrams," on page 139. This parameter is a mandatory part of every XML message. It must also be unique among all calls that are pending at the server. It is the client's responsibility to guarantee this uniqueness, even when multiple clients exist for the same IVR Server. In parameter tables beginning on Page 159, literal strings are indicated in bold whereas mapped values are indicated with italics.

This section does not demonstrate actual XML message structure. Parameter, as a term, is used to indicate either an attribute or entity related to the message entity. Refer to Appendix, "The IVR Server DTD," on page 173for structural information.

Messages that exist in the DTD that are not detailed here either:

- Do not pertain to Network mode—for example, the `MakeCall` message.
- Are meaningless when used in Network mode— for example, the `LoginReq` message.

New Call and Call-Routing Messages

These messages are used to start a call, route it, confirm the connection or indicate failure to connect, and end the call.

NewCall

Sent from the IVR to IVR Server, this message notifies the server that a call has arrived at the network platform. This message establishes the call ID that should will be used throughout the call. `CalledNum` may not have its entirety mapped to `AttributeThisDN`. A configuration option may specify that a certain number of prefix digits be removed. Although "`Version`" is optional, it defaults to "`1.0`". It is recommended that all calls use "`2.0`" or higher. Using version "`1.0`" will disable the list structure used in `ExtnsEx` and `UDataEx`. The deprecated forms of these structures are not covered in this document.

See Table 58 for a complete list of message parameters.

Table 58: NewCall Message Parameters

Message	Direction	Parameter		Optional/ Required
		Name	Value	
NewCall	IVR to IVR Server	CallControlMode	Genesys Network	Required
		CalledNum	AttributeThisDN	Required
		Version	1.0 2.0 3.0 4.0	Optional
		ANIRestriction	CLIP—(Calling Line Identification Presentation) CLIR—(Calling Line Identification Restriction)	Reserved for Future Use
		DNIS	AttributeDNIS	Optional
		ANI	AttributeANI	Optional
		UDataEx	AttributeExtensions	Optional
		ExtnsEx	AttributeUserData	Optional

RouteRequest

Sent from the IVR to IVR Server, this message informs IVR Server that a call started using `NewCall` (`CallControlMode=Network`) requires instructions from Universal Routing Server. The `RouteDN` attribute listed in the DTD is ignored in network mode. Route requests will be posted against the DN specified in `NewCall`.

See Table 59 for a complete list of message parameters.

Table 59: RouteRequest Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
RouteRequest	IVR to IVR Server	CED	AttributeCollectedDigits	Optional
		UDataEx	AttributeExtensions	Optional
		ExtnsEx	AttributeUserData	Optional

RouteResponse

Sent from the IVR Server to the IVR, this message notifies the client that the call should be routed to the specified destination. `Dest` will be present in cases where `RouteType` is not `Default`. It may or may not be present when default routing is requested.

See Table 60 for a complete list of message parameters.

Table 60: RouteResponse Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
RouteResponse	IVR Server to IVR	RouteType	Default Normal Reroute RerouteAttended RerouteConferenced	Required
		Dest	AttributeOtherDN	Optional
		ExtnsEx	AttributeExtensions	Optional

Connected

Sent from the IVR to IVR Server, this message indicates that the call has been delivered to the destination specified in the previous `RouteResponse` message from IVR Server. When this message is received by IVR Server, the corresponding Universal Routing Server strategy will be ended.

See Table 61 for a complete list of message parameters.

Table 61: Connected Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
Connected	IVR to IVR Server	ExtnsEx	AttributeExtensions	Optional

EndCall

Sent from the IVR to IVR Server or from the IVR Server to the IVR, this message indicates that a call has ended. If the client receives this message from the server, no expectation should be made regarding outstanding requests on the related call. An `EndCause` of `Timeout` should be interpreted as a default handling instruction when sent by IVR Server. A cause of `FeatureNotSupported` indicates a configuration error.

See Table 62 for a complete list of message parameters.

Table 62: EndCall Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
EndCall	IVR Server to IVR or IVR to IVR Server	EndCause	Abandoned FeatureNotSupported InvalidStateTransition InvalidVersion Normal Resources Timeout	Required
		UDataEx	AttributeUserData	Optional
		ExtnsEx	AttributeExtensions	Optional

Failure

Sent from the IVR to IVR Server, after receiving routing instructions, this message indicates that the routing operation failed. This will result in an `EventError` being returned to Universal Routing Server.

See Table 63 for a complete list of message parameters.

Table 63: Failure Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
Failure	IVR to IVR Server	FailureCause	Busy ConnectionFailed NoAnswer	Required
		ExtnsEx	AttributeExtensions	Optional

The error code used when sending `EventError` to Universal Routing Server will be determined based upon the value of `FailureCause`. Those particular error codes are listed in Table 64.

Table 64: Failure Error Codes

FailureCause	T-Library Error Value	AttributeErrorCode
Busy	TERR_ORIG_DN_BUSY	83
NoAnswer	TERR_DN_NO_ANSWER	232
ConnectFailed	TERR_CONN_ATMPT_FIL	234

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

Sent from the IVR Server to the IVR, in response to a `RouteRequest`, this message notifies the client that the call should receive the specified treatment. The `Type` parameter is converted from the T-Library `AttributeTreatmentType` enumeration.

See Table 65 for a complete list of message parameters.

Table 65: TreatCall Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
TreatCall	IVR Server to IVR	Type	Busy CancelCall CollectDigits DeleteAnnounce FasyBusy IVR Music PlayAnnounce PlayAnnounceAndDigits PlayApplication RAN RecordAnnouce RingBack SetDefaultRoute Silence TextToSpeech TextToSpeechAndDigits VerifyDigits	Required
		ExtnsEx	AttributeExtensions	Optional
		Parameters	AttributeTreatmentParms	Optional

TreatStatus

Sent from the IVR to the IVR Server, this message informs the server of the progress of a previous treatment request.

See Table 66 for a complete list of message parameters.

Table 66: TreatStatus Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
TreatStatus	IVR to IVR Server	Status	`Started` NotStarted Completed	Required
		CED	AttributeCollectedDigits	Optional
		ExtnsEx	AttributeExtensions	Optional
		UDataEx	AttributeUserData	Optional

Cancel

Sent from the IVR Server to the IVR, this message notifies the client that a currently running treatment should be canceled. This message is only sent when the cancellation is due to a routing request. This message has no parameters.

CancelCompleted

Sent from the IVR to the IVR Server, this is the proper response to the `Cancel` message, this indicates that the treatment has been cancelled and the network is ready for routing instructions. This message has no parameters.

Call Information Messages

These messages request data attached to the call and return the corresponding response.

CallInfoReq

Sent from the IVR to IVR Server, this message requests that call information be returned to the client. This message has no parameters.

CallInfoResp

Sent from the IVR Server to the IVR, in response to `CallInfoReq`, this message contains information related to the listed parameters that have corresponding data. This information is specifically related to T-Library side attributes, though in network mode these values are often determined by attributes of the `NewCall` message.

In cases where a call is routed to IVR Server using route type external routing, the `PortDN` field will contain the value of `AttributeThisDN` after being moved to the route target.

The `FirstHomeLocation` field will only be present when the call using the version 3 protocol. See "NewCall" on page 158 for more information.

See Table 67 for a complete list of message parameters.

Table 67: CallInfoResp Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
CallInfoResp	IVR Server to IVR	ANI	AttributeANI	Optional
		CalledNum	AttributeThisDN	Optional
		ConnId	AttributeConnID	Optional
		DNIS	AttributeDNIS	Optional
		FirstHomeLocation	See details above	Optional
		LastEvent	N/A	Optional
		OtherDN	AttributeOtherDN	Optional
		OtherQueue	AttributeOtherQueue	Optional
		OtherTrunk	AttributeOtherTrunk	Optional
		PortDN	See details above	Optional
		PortQueue	AttributeThisQueue	Optional
		PortTrunk	AttributeThisTrunk	Optional
		TSCallId	AttributeCallID	Optional

Statistics Messages

The statistics messages enable you to request and 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.

GetStatReq

Sent from the IVR to the IVR Server, this message requests information for a specified statistic. Statistics cannot be arbitrarily requested, and they must be configured fully in Configuration Manager prior to use. This request is thus functionally equivalent to the "PeekStatReq" on page 167. The `RequestId` field must be unique for all outstanding statistic requests on a given call. This value will be returned in the subsequent response.

See Table 68 for a complete list of message parameters.

Table 68: GetStatReq Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
GetStatReq	IVR to IVR Server	ObjectId	obj_id (from Configuration Manager statistic definition)	Required
		ObjectType	obj_type (from Configuration Manager statistic definition)	Required
		RequestId	Client-determined reference ID	Required
		ServerName	server_name (from Configuration Manager statistic definition)	Required
		StatType	stat_type (from Configuration Manager statistic definition)	Required

PeekStatReq

Sent from the IVR to the IVR Server, this message requests information for a specified statistic by specifying the Configuration Manager statistic ID.

See Table 69 for a complete list of message parameters.

Table 69: PeekStatReq Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
PeekStatReq	IVR to IVR Server	StatName	Name of Configuration Manager statistic definition's	Required
		RequestId	Client-determined reference ID	Required

StatResp

Sent from the IVR Serve to the IVR, this message is the response for a statistic requests, peek or get.

See Table 70 for a complete list of message parameters.

Table 70: StatResp Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
StatResp	IVR Server to IVR	RequestId	Reference ID from the original request	Required
		ResultCode	MiscError NoSuchStat Success	Required
		Result	StatServer reported value	Optional

User Data Messages

These messages enable you to access and control data about the actions performed by callers.

UDataGet

Sent from the IVR to the IVR Server, this message requests values for the specified keys from the call's user data. The keys field is a colon delimited string indicating all required keys.

See Table 71 for a complete list of message parameters.

Table 71: UDataGet Message parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
UDataGet	IVR to IVR Server	Keys	key1[:key2[keyn]]	Required
		RequestId	Client-determined reference ID	Required

UDataGetAll

Sent from the IVR to the IVR Server, this message requests values for the all keys present in the call's user data.

See Table 72 for a complete list of message parameters.

Table 72: UDataGetAll Message parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
UDataGetAll	IVR to IVR Server	RequestId	Client-determined reference ID	Required

UDataResp

Sent from the IVR Server to the IVR, this message is the response message for user data requests. It indicates failure or success as well as any relevant results, which will be in the `UDataEx` field.

See Table 73 for a complete list of message parameters.

Table 73: UDataResp Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
UDataResp	IVR Server to IVR	RequestId	Reference ID from the original request	Required
		Result	FeatureNotSupported MiscError NoMatch NoSuchCall Success	Required
		UDataEx	Matching user data, when in response to `UDataGet`, `UDataGetAll`	Optional

UDataSet

Sent from the IVR to the IVR Server, this message allows the client to update the T-Library user data for the associated call. Only the `Replace` operation is supported for network clients. This operation will nevertheless add a new key/value pair, if it doesn't currently exist. The `UDataEx` only need to indicate user data which is new, this does not affect existing keys in the T-Library user data.

See Table 74 for a complete list of message parameters.

Table 74: UDataSet Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
UDataSet	IVR to IVR Server	Action	Add Replace	Required
		RequestId	Client-determined reference ID	Required
		UDataEx	AttributeUserData	Required

Transfer/Conferencing Messages

These messages are used to control call transfers and conferencing.

CallError

Sent from the IVR Server to the IVR, this message is sent whenever a message received from the client causes an error. These errors can result from failure to follow the prescribed call model or from remote errors caused by a particular request. In the case of remote failure, the T-Library error code will be provided for reference.

The `FailedReq` parameter can only be one of the two values:

- `NoSuchCall`
- `Unknown`

Values not listed here that exist in the DTD file apply to other operational modes.

See Table 75 for a complete list of message parameters.

Table 75: CallError Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
CallError	IVR Server to IVR	FailedReq	NoSuchCall Unknown	Required
		TLibErrorCode	AttributeErrorCode (where applicable)	Optional

General Messages

This message is used for logging.

LogMsg

Sent from the IVR to the IVR Server, this message allows the client application to write messages into IVR Server logs, at the specified logging level.

See Table 76 for a complete list of message parameters.

Table 76: LogMsg Message Parameters

Message	Direction	Parameter		Optional/Required
		Name	Value	
CallError	IVR to IVR Server	MsgType	Debug Standard Trace	Required
		Msg	Message to be logged	Required

Genesys

Appendix

The IVR Server DTD

This appendix includes the entire text of version 4.0 of the `IServer.dtd` file.

```
<?xml encoding="ISO-8859-1"?>

<!-- Copyright (c) 2001 - 2006 Genesys Telecommunications               -->
<!-- Laboratories, Inc.  All rights reserved.                           -->


<!ELEMENT GctiMsg (
     (CallId, (NewCall       | RouteRequest    | RouteResponse     |
               Connected     | EndCall          | Failure           |
               Cancel        | CancelCompleted | Reset             |
               PeekStatReq   | GetStatReq       | StatResp          |
               LoginReq      | LoginResp        | LogMsg            |
               UDataGet      | UDataGetAll      | UDataSet          |
               UDataDel      | UDataResp        | CallInfoReq       |
               CallInfoResp  | AccessNumGet     | AccessNumCancel   |
               AccessNumResp | OneStepXfer      | OneStepConf       |
               InitConf      | CompleteConf     | InitXfer          |
               CompleteXfer  | RetrieveCall     | CallStatus        |
               CallError     | MakeCall         | TreatCall         |
               TreatStatus   | DialOutInit      | DialOutRegistry   |
               FlowControl   | DialOutRegistryResp )
     )         |
     DialOut   |
     DialOutError
)>

<!ELEMENT CallId  (#PCDATA)>


<!-- ====================== Outbound Dialing ================== -->
<!ELEMENT DialOut      (UDataEx?, ExtnsEx?)>
<!ATTLIST DialOut       RefIDCDATA #REQUIRED
        OrigNumCDATA #REQUIRED
                       DestNumCDATA #REQUIRED
        TimeToAnswerCDATA #REQUIRED>
```

```
<!ELEMENT DialOutError EMPTY>
<!ATTLIST DialOutError Error  (NotSupported | NoTrunks | MiscError) #REQUIRED
                       RefID  CDATA                                #REQUIRED>


<!ELEMENT DialOutRegistry EMPTY>
<!ATTLIST DialOutRegistry Command (Add | Remove | RemoveAll ) #REQUIRED
                          DN       CDATA                     #IMPLIED>

<!ELEMENT DialOutRegistryResp EMPTY>
<!ATTLIST DialOutRegistryResp Result
                         (MiscFailure | ConfigError | Success) #REQUIRED>


<!ELEMENT DialOutInit    (CalledNum, ExtnsEx?)>
<!ATTLIST DialOutInit     RefID    CDATA      #REQUIRED
                          Version (2.0 | 3.0) "3.0">


<!-- ====================== Call Control ======================= -->

<!ELEMENT NewCall     (CalledNum, DNIS?, ANI?,
                       ((UserData?, Extensions?) | (UDataEx?, ExtnsEx?)))>

<!ATTLIST NewCall      CallControlMode (Genesys | Network) #REQUIRED
                       Version         (1.0  | 2.0 | 3.0 | 4.0)  "1.0"
                       ANIRestriction  (CLIP | CLIR) #IMPLIED>

<!ELEMENT MakeCall    (UDataEx?, ExtnsEx?)>
<!ATTLIST MakeCall     OrigNum  CDATA #REQUIRED
                       DestNum  CDATA #REQUIRED
                       Location CDATA #IMPLIED>

<!ELEMENT CalledNum   (#PCDATA)>
<!ELEMENT DNIS        (#PCDATA)>
<!ELEMENT ANI         (#PCDATA)>
<!ELEMENT UserData    (NVPair)+>
<!ELEMENT Extensions  (NVPair)+>
<!ELEMENT NVPair      (NVName, NVVal)>
<!ELEMENT NVName      (#PCDATA)>
<!ELEMENT NVVal       (#PCDATA)>

<!ELEMENT RouteRequest (CED?,
                        ((UserData?, Extensions?) | (UDataEx?, ExtnsEx?)))>
<!ATTLIST RouteRequest  RouteDN CDATA #IMPLIED>

<!ELEMENT CED (#PCDATA)>

<!ELEMENT RouteResponse (Dest?, (Extensions | ExtnsEx)?)>
<!ATTLIST RouteResponse  RouteType (Default | Normal         |
```

```
                                  Reroute |RerouteAttended |
                                  RerouteConferenced) #REQUIRED>
<!ELEMENT Dest (#PCDATA)>

<!ELEMENT Connected ((Extensions | ExtnsEx)?)>

<!ELEMENT EndCall (((UserData?, Extensions?) | (UDataEx?, ExtnsEx?)))>
<!ATTLIST EndCall    EndCause (Normal | Abandoned     | Resources      |
                               FeatureNotSupported    | InvalidVersion |
                               InvalidStateTransition | Timeout        |
                               ServerPaused              ) #REQUIRED>

<!ELEMENT Failure ((Extensions | ExtnsEx)?)>
<!ATTLIST Failure FailureCause (Busy | NoAnswer | ConnectFailed) #REQUIRED>

<!-- UUI_Number is only used for DirectUUI type external routing.    -->
<!-- The value of this parameter must be a 32-bit integer that will  -->
<!-- be passed by the IVR system to the destination switch using     -->
<!-- ISDN User to User Information signalling.                       -->
<!-- The value 'DirectAniDnis' for attribute XRouteType is           -->
<!-- reserved for future use.                                        -->
<!ELEMENT AccessNumGet (UDataEx?, ExtnsEx?)>
<!ATTLIST AccessNumGet DestDN      CDATA                          #REQUIRED
                       Location    CDATA                          #REQUIRED
                       XRouteType (Default        | Route         |
                                   Reroute        | Direct        |
                                   DirectAni      | DirectNotoken |
                                   DirectAniDnis  | DirectUUI     |
                                   DirectDigits   | DnisPool)       "Default"
                       UUI_Number  CDATA                          #IMPLIED>

<!ELEMENT AccessNumResp EMPTY>
<!ATTLIST AccessNumResp Action    (Get     | Cancel)  #REQUIRED
                         Result    (Success | Failure) #REQUIRED
                         AccessNum  CDATA              #IMPLIED>


<!ELEMENT AccessNumCancel EMPTY>

<!ELEMENT CallStatus EMPTY>
<!ATTLIST CallStatus Event (Dialing       | Ringing | Established  |
                            Retrieved     | Busy    | Held         |
                            ConfPartyAdd  | ConfPartyDel           |
                            XferComplete  | Released) #REQUIRED>




<!ELEMENT CallError EMPTY>
<!ATTLIST CallError FailedReq  (Unknown | NoSuchCall |
                                OneStepXfer | OneStepConf |
```

```
                              InitConf | CompleteConf |
                              InitXfer | CompleteXfer |
                              MakeCall | RetrieveCall |
                              AgentControl | NotAllowed )  #REQUIRED
                    TlibErrCode CDATA                       #IMPLIED
                    ReqId       CDATA                       #IMPLIED >


<!ELEMENT OneStepXfer     (UDataEx?, ExtnsEx?)>
<!ATTLIST OneStepXfer      DestDN      CDATA              #REQUIRED
                           Location    CDATA              #IMPLIED>

<!ELEMENT OneStepConf     (UDataEx?, ExtnsEx?)>
<!ATTLIST OneStepConf      DestDN      CDATA              #REQUIRED
                           Location    CDATA              #IMPLIED>

<!ELEMENT InitConf        (UDataEx?, ExtnsEx?)>
<!ATTLIST InitConf         DestDN      CDATA              #REQUIRED
                           Location    CDATA              #IMPLIED>
<!ELEMENT CompleteConf    (ExtnsEx?)>

<!ELEMENT InitXfer        (UDataEx?, ExtnsEx?)>
<!ATTLIST InitXfer         DestDN      CDATA              #REQUIRED
                           Location    CDATA              #IMPLIED>
<!ELEMENT CompleteXfer    (ExtnsEx?)>

<!ELEMENT RetrieveCall    (ExtnsEx?)>

<!-- ====================== Call Info Retrieval ================== -->

<!ELEMENT CallInfoReq  EMPTY>
<!ATTLIST CallInfoReq  ReportUUID (true | false) #IMPLIED ReportThirdPartyDN (true |
   false) #IMPLIED >
<!ELEMENT CallInfoResp EMPTY>
<!ATTLIST CallInfoResp  ANI               CDATA #IMPLIED
                        DNIS              CDATA #IMPLIED
                        CalledNum         CDATA #IMPLIED
                        ConnId            CDATA #IMPLIED
                        TSCallId          CDATA #IMPLIED
                        PortDN            CDATA #IMPLIED
                        PortTrunk         CDATA #IMPLIED
                        PortQueue         CDATA #IMPLIED
                        OtherDN           CDATA #IMPLIED
                        OtherTrunk        CDATA #IMPLIED
                        OtherQueue        CDATA #IMPLIED
                        LastEvent         CDATA #IMPLIED
                        FirstHomeLocation CDATA #IMPLIED
                        UUID              CDATA #IMPLIED
                        ThirdPartyDN       CDATA #IMPLIED>
```

```
<!-- ======================= Call Treatments ======================= -->

<!-- Parameters are present for all treatment types except           -->
<!-- RingBack, Silence, Busy and CancelCall                          -->
<!ELEMENT TreatCall (Parameters?, ExtnsEx?)>
<!ATTLIST TreatCall Type (PlayAnnounce    | PlayAnnounceAndDigits            |
                          Music           | RAN            | Busy            |
                          CollectDigits   | CancelCall     | SetDefaultRoute |
                          PlayApplication | IVR            | RingBack        |
                          Silence         | VerifyDigits   | RecordAnnounce  |
                          DeleteAnnounce  | TextToSpeech   | FastBusy        |
                          TextToSpeechAndDigits) #REQUIRED>

<!ELEMENT Parameters (Node | List)+>

<!ELEMENT TreatStatus (UDataEx?, ExtnsEx?)>
<!ATTLIST TreatStatus Status (Started | NotStarted | Completed) #REQUIRED
                      CED     CDATA                             #IMPLIED>

<!ELEMENT Cancel          EMPTY>
<!ELEMENT CancelCompleted EMPTY>


<!-- ======================= Tlib Proxy ==================== -->

<!ELEMENT MonitorInfo (Server | Port | Agent )>
<!ATTLIST MonitorInfo  ReqId  CDATA  #IMPLIED >

<!ELEMENT Server EMPTY>
<!ATTLIST Server Name    CDATA                       #REQUIRED
                 Status (OK | Unavailable)          #REQUIRED
       Switch  CDATA                      #IMPLIED>

<!ELEMENT Port EMPTY>
<!ATTLIST Port PortNum  CDATA             #REQUIRED
               Status  (OK | Unavailable) #REQUIRED>


<!ELEMENT Agent EMPTY>
<!ATTLIST Agent PortNum  CDATA                             #REQUIRED
                Status  (LoggedIn | LoggedOut | Ready |
                         NotReady | Unknown)              #REQUIRED>

<!ELEMENT AgentQuery EMPTY>
<!ATTLIST AgentQuery ReqId    CDATA  #REQUIRED
                     PortNum  CDATA  #REQUIRED >

<!ELEMENT AgentLogin EMPTY>
<!ATTLIST AgentLogin ReqId     CDATA  #REQUIRED
                     PortNum   CDATA  #REQUIRED
                     Queue     CDATA  #REQUIRED
```

```
                    AgentId   CDATA  #REQUIRED
                    Password  CDATA  #REQUIRED >

<!ELEMENT AgentLogout EMPTY>
<!ATTLIST AgentLogout ReqId     CDATA  #REQUIRED
                      PortNum   CDATA  #REQUIRED
                      Queue     CDATA  #REQUIRED >

<!ELEMENT AgentReady EMPTY>
<!ATTLIST AgentReady ReqId     CDATA                          #REQUIRED
                     PortNum   CDATA                          #REQUIRED
                     Queue     CDATA                          #REQUIRED
                     WorkMode (AutoIn | ManualIn | Unknown) "Unknown" >

<!ELEMENT AgentNotReady EMPTY>
<!ATTLIST AgentNotReady ReqId     CDATA                     #REQUIRED
                        PortNum   CDATA                     #REQUIRED
                        Queue     CDATA                     #REQUIRED
                        WorkMode (AutoIn | ManualIn |
                                  Unknown)                  "Unknown" >


<!-- ======================== Misc ==================== -->

<!ELEMENT LoginReq   EMPTY>
<!ATTLIST LoginReq   Version        (1.0 | 2.0 | 3.0 | 4.0) #REQUIRED
                     ClientName      CDATA                   #REQUIRED
                     ReportStatus   (true | false)           #IMPLIED
          ServerMonitor  (set | clear)           #IMPLIED>

<!ELEMENT LoginResp (ConfigOptions?)>
<!ATTLIST LoginResp  IServerVer CDATA                              #REQUIRED
                     Result (Success | InvalidProtocolVer)         #REQUIRED
                     Status (NoSuchClient | InitInProgress | OK) #IMPLIED>

<!ELEMENT FlowControl EMPTY>
<!ATTLIST FlowControl Status (On | Off) #REQUIRED>

<!ELEMENT LogMsg EMPTY>
<!ATTLIST LogMsg MsgType (Standard | Trace | Debug) #REQUIRED
                 Msg      CDATA                     #REQUIRED>

<!ELEMENT Reset ((Extensions | ExtnsEx)?)>

<!-- ========================= Statistics ====================== -->

<!ELEMENT PeekStatReq (RequestId, StatName)>
<!ELEMENT GetStatReq  (RequestId, ServerName, StatType, ObjectId, ObjectType)>
<!ELEMENT StatResp    (RequestId, Result?)>
<!ATTLIST StatResp     ResultCode
                        (Success | NoSuchStat | MiscError) #REQUIRED>
```

```
<!ELEMENT RequestId  (#PCDATA)>
<!ELEMENT StatName   (#PCDATA)>
<!ELEMENT ServerName (#PCDATA)>
<!ELEMENT StatType   (#PCDATA)>
<!ELEMENT ObjectId   (#PCDATA)>
<!ELEMENT ObjectType (#PCDATA)>

<!ELEMENT Result       EMPTY>
<!ATTLIST Result Value CDATA #REQUIRED>

<!-- ====================== User Data Management ================== -->
<!ELEMENT UDataGetAll (RequestId)>

<!-- The 'Keys' attribute is a colon separated list of key names to  -->
<!-- retrieve.                                                       -->

<!ELEMENT UDataGet (RequestId)>
<!ATTLIST UDataGet Keys CDATA #REQUIRED>

<!-- The UDataEx list will only be present when UDataResp is          -->
<!-- sent in reply to a UdataGet, and will contain the user data      -->
<!-- items that could be successfully retrieved.  In case none of the -->
<!-- supplied key names existed in UserData the UDataEx list will     -->
<!-- not be present and the 'Result' attribute will be set to NoMatch -->

<!ELEMENT UDataResp (RequestId, UDataEx?)>
<!ATTLIST UDataResp Result (Success | NoSuchCall | NoMatch |
                            FeatureNotSupported  | MiscError) #REQUIRED>

<!-- If the 'Action' attribute is Add the data in the UDataEx        -->
<!-- will be added to the list, possibly creating duplicate entries  -->
<!-- with the same key name.  If the 'Action' attribute is Replace   -->
<!-- then the data in UDataEx will overwrite any existing entries    -->
<!-- that are already be present in UserData.                        -->

<!ELEMENT UDataSet (RequestId, UDataEx)>
<!ATTLIST UDataSet Action (Add | Replace) #REQUIRED>

<!-- If the 'Action' attribute is DeleteAll then all user data will be -->
<!-- deleted and the 'Key' attribute need not be present.  If the      -->
<!-- 'Action' attribute is DeleteKey then only attribute 'Key' will be -->
<!-- deleted                                                           -->

<!ELEMENT UDataDel (RequestId)>
<!ATTLIST UDataDel Action (DeleteAll | DeleteKey) #REQUIRED
                   Key CDATA #IMPLIED>



<!-- ====================== Typed, Nested List ================== -->
```

```
<!ELEMENT List (Node | List)+>
<!ATTLIST List  Name CDATA #REQUIRED>

<!ELEMENT Node  EMPTY>
<!ATTLIST Node  Name  CDATA            #REQUIRED
                Type (Int | Str | Bin) #REQUIRED
                Val   CDATA            #REQUIRED>

<!ELEMENT UDataEx       (Node | List)+>
<!ELEMENT ExtnsEx       (Node | List)+>
<!ELEMENT ConfigOptions (Node)+>
```

Genesys

Supplements

Related Documentation Resources

The following resources provide additional information that is relevant to this software. Consult these additional resources as necessary.

IVR Interface Option

- A series of *IVR Driver System Administrator's Guide*, which provide information about how to install and configure specific IVR drivers supplied by third-party vendors.
- *The IVR Interface Option 8.5 IVR Server System Administrator's Guide,* which will help you understand the architecture of the Genesys IVR product.
- Release Notes and Product Advisories for this product, which are available on the Genesys Documentation website.

Genesys

- *Genesys Technical Publications Glossary,* which provides a comprehensive list of the Genesys and computer-telephony integration (CTI) terminology and acronyms used in this document.
- *Genesys Migration Guide*, which provides documented migration strategies for Genesys product releases. Contact Genesys Customer Care for more information.

Information about supported operating systems and third-party software is available on the Genesys Documentation website in the following documents:

- *Genesys Supported Operating Environment Reference Guide*
- *Genesys Supported Media Interfaces Reference Manual*

Consult the following additional resources as necessary:

- *Genesys Hardware Sizing Guide,* which provides information about Genesys hardware sizing guidelines for the Genesys 8.x releases.

- *Genesys Interoperability Guide,* which provides information on the compatibility of Genesys products with various Configuration Layer Environments; Interoperability of Reporting Templates and Solutions; and Gplus Adapters Interoperability.
- *Genesys Licensing Guide,* which introduces you to the concepts, terminology, and procedures that are relevant to the Genesys licensing system.
- *Genesys Database Sizing Estimator 8.x Worksheets,* which provides a range of expected database sizes for various Genesys products.

For additional system-wide planning tools and information, see the release-specific listings of System-Level Documents on the Genesys Documentation website.

Genesys product documentation is available on the:

- Genesys Customer Care website.
- Genesys Documentation website.
- Genesys Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesys.com.

Document Conventions

This document uses certain stylistic and typographical conventions—introduced here—that serve as shorthands for particular kinds of information.

Document Version Number

A version number appears at the bottom of the inside front cover of this document. Version numbers change as new information is added to this document. Here is a sample version number:

85fr_ref_06-2014_v8.5.001.00

You will need this number when you are talking with Genesys Customer Care about this product.

Screen Captures Used in This Document

Screen captures from the product graphical user interface (GUI), as used in this document, may sometimes contain minor spelling, capitalization, or grammatical errors. The text accompanying and explaining the screen captures corrects such errors *except* when such a correction would prevent you from installing, configuring, or successfully using the product. For example, if the name of an option contains a usage error, the name would be presented exactly as it appears in the product GUI; the error would not be corrected in any accompanying text.

Type Styles

Table 77 describes and illustrates the type conventions that are used in this document.

Table 77: Type Styles

Type Style	Used For	Examples
Italic	• Document titles • Emphasis • Definitions of (or first references to) unfamiliar terms • Mathematical variables Also used to indicate placeholder text within code samples or commands, in the special case where angle brackets are a required part of the syntax (see the note about angle brackets on page 184).	Please consult the *Genesys Migration Guide* for more information. Do *not* use this value for this option. A *customary and usual* practice is one that is widely accepted and used within a particular industry or profession. The formula, $x + 1 = 7$ where x stands for . . .
Monospace font (Looks like `teletype` or `typewriter text`)	All programming identifiers and GUI elements. This convention includes: • The *names* of directories, files, folders, configuration objects, paths, scripts, dialog boxes, options, fields, text and list boxes, operational modes, all buttons (including radio buttons), check boxes, commands, tabs, CTI events, and error messages. • The values of options. • Logical arguments and command syntax. • Code samples. Also used for any text that users must manually enter during a configuration or installation procedure, or on a command line.	Select the `Show variables on screen` check box. In the `Operand` text box, enter your formula. Click `OK` to exit the `Properties` dialog box. T-Server distributes the error messages in `EventError` events. If you select `true` for the `inbound-bsns-calls` option, all established inbound calls on a local agent are considered business calls. Enter `exit` on the command line.
Square brackets ([ ])	A particular parameter or value that is optional within a logical argument, a command, or some programming syntax. That is, the presence of the parameter or value is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information.	`smcp_server -host [/flags]`
Angle brackets (< >)	A placeholder for a value that the user must specify. This might be a DN or a port number specific to your enterprise. **Note:** In some cases, angle brackets are required characters in code syntax (for example, in XML schemas). In these cases, italic text is used for placeholder values.	`smcp_server -host <confighost>`

Index

Symbols

A

B

C

D

E

F

G

I

K

L

M

N

O

P

R

S

T

U

V

X