This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Agent Interaction SDK Java Developer Guide

Voice Sequence Diagrams

12/14/2025

Contents

Voice Sequence Diagrams

This appendix presents sequence diagrams for voice interactions.

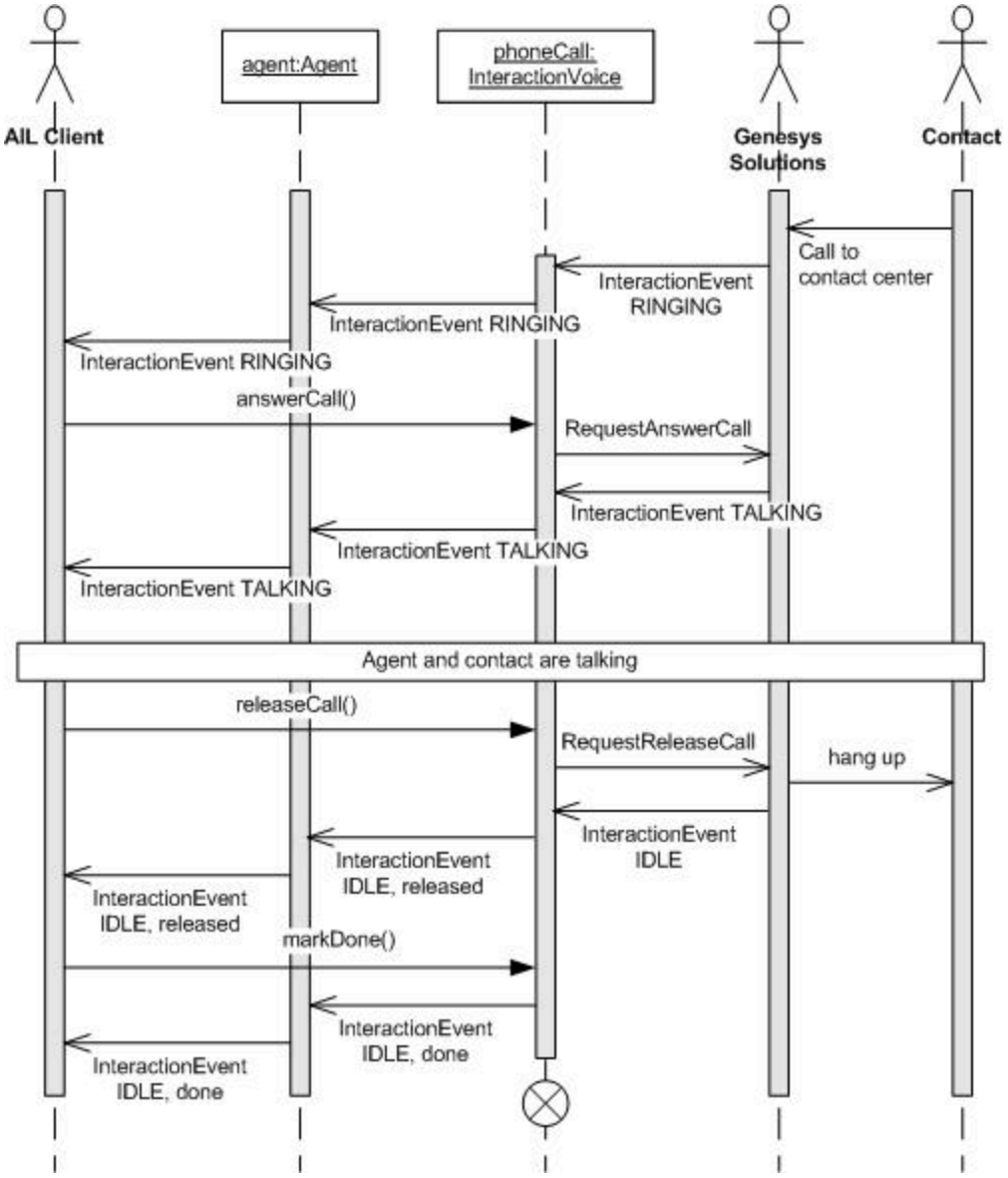
Make a Phone Call

The first step in making a phone call is to create an interaction of type `VOICE`. You do this by calling the `createInteraction()` method on your `Agent` or`Place` interface with the type of interaction. As a result, your application gets an `InteractionEvent` for a voice interaction in `NEW` status.

Make the phone call by invoking the `makeCall()` method on the `InteractionVoice` interface. At this point, the interaction status becomes `DIALING`, as specified in the corresponding `InteractionEvent`. When the connection is established, during the call, the interaction is in `TALKING` status.

Either you initiate the hang up by calling the `releaseCall()` method, or the peer has hung up. Either way, you receive an `InteractionEvent` event of type `IDLE`.

You finish and clean up the interaction by calling the `markDone()` method, which releases any reference to the interaction in the library. It also saves the interaction in the history, if you have established a connection between the library and the Contact Server database.
Event flow for making a voice call is shown in Making a Call.



Making a Call

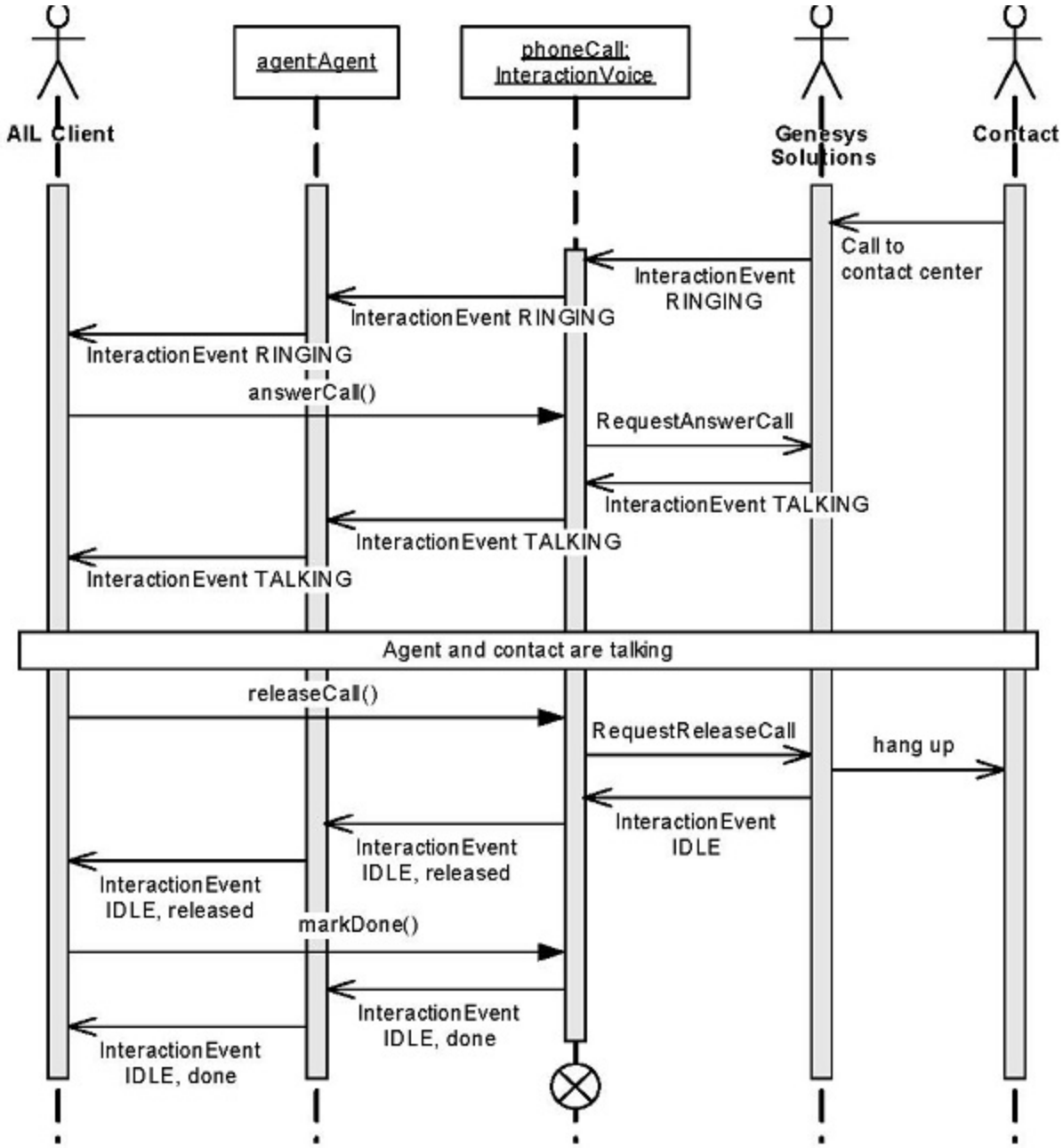
Answer a Phone Call

You have to answer a phone call when you get an `InteractionEvent` for a voice interaction in `RINGING` status. To answer the phone call, use the `InteractionVoice.answerCall()` method. As a result of the `ANSWER` action, you receive the event indicating that the interaction is in status `TALKING`.

The end of the interaction is the same as making a phone call: you hang up with `InteractionVoice.releaseCall()` method and you finish with the `InteractionVoice.markDone()`

method.

The event flow for answering a call is shown below.



Answering a Call

Conferencing

A conferencing scenario can be divided into three steps:

1. A Contact calls a first agent, named agent1.
2. agent1 initiates a conference with a second agent, named agent2.
3. agent1 creates the conference.

In the first step, agent1 receives a phone call. It is exactly the same scenario as Answer a Phone Call:

- agent1 receives an InteractionEvent event carrying an InteractionVoice interface, named phoneCall1, with the status RINGING.
- agent1 takes the call by invoking the answerCall() method.

In the second step, agent1, already in communication with the contact, invokes the initiateConference() method on the InteractionVoice interface phoneCall1 to prepare the conference with agent2.

After this call, agent1 receives an InteractionEvent event setting the InteractionVoice phoneCall1 to the status HELD. The communication with the contact is paused.

The Agent Interaction Layer creates then two InteractionVoice core objects:

- One for agent1, setting an interaction with agent2. This interaction follows the scenario in Make a Phone Call, and is named phoneCall2.
- The other for agent2, representing his communication first with agent1 and then with the conference. This interaction follows the scenario in Answer a Phone Call and is named phoneCall3.

Next, agent1 receives an InteractionEvent event carrying the newly created InteractionVoice phoneCall2 with the status DIALING. Simultaneously, agent2 receives an InteractionEvent event carrying its InteractionVoice phoneCall3 with the status RINGING. When agent2 answers the call by invoking the answerCall() method, both agents receive an InteractionEvent event showing that their InteractionVoice core objects have the status TALKING. They are now in communication with each other.

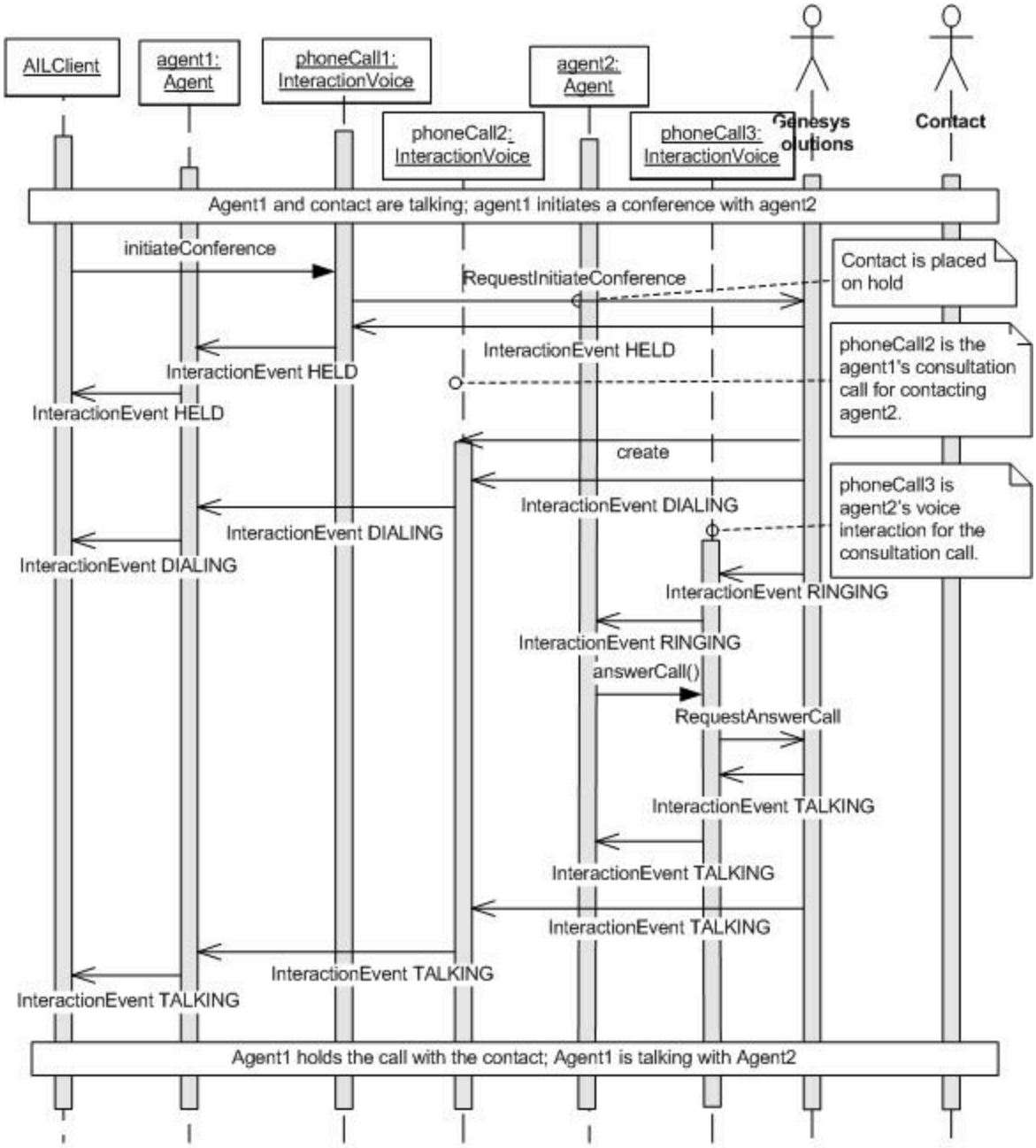
In the third step, when the two agents are ready to proceed, agent1 invokes the completeConference() method.

Next, agent1 receives an InteractionEvent event setting the status of the interaction referred by phoneCall2 to IDLE, then this interaction is destroyed.
Finally, agent1 receives two successive InteractionEvent events:

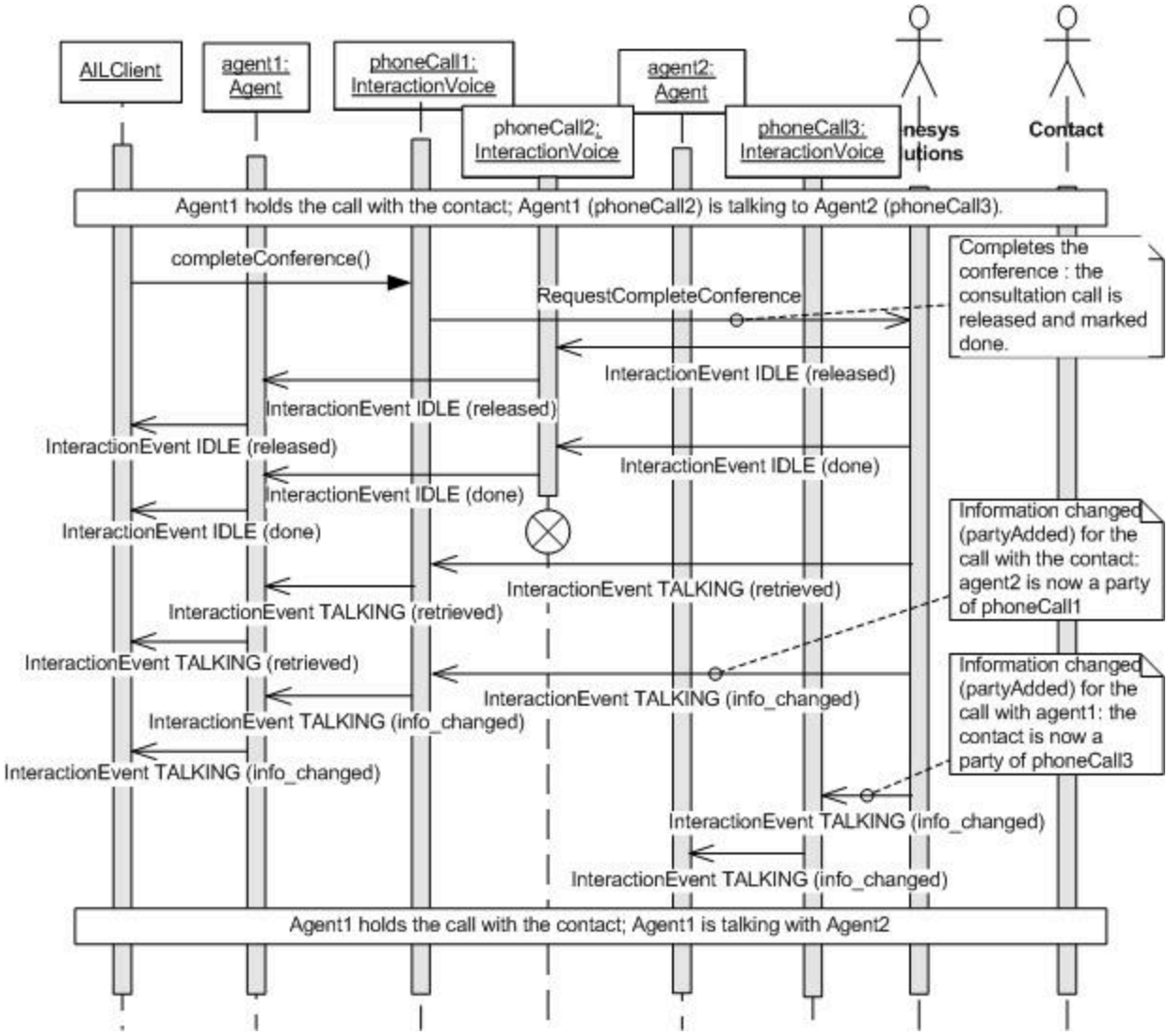
- One resuming the interaction phoneCall1 with the contact, setting it to the status TALKING.
- Another to notify the arrival of a third peer, agent2, on phoneCall1.

Agent2 also receives an InteractionEvent event notifying it of the arrival of a third peer, the contact, on its InteractionVoice phoneCall3.

The conference can now take place, and for each of the agents, each interaction is viewed as a standard phone call and must be ended accordingly.
The event flow is presented in the following two figures.



Call Conferencing, Initiating

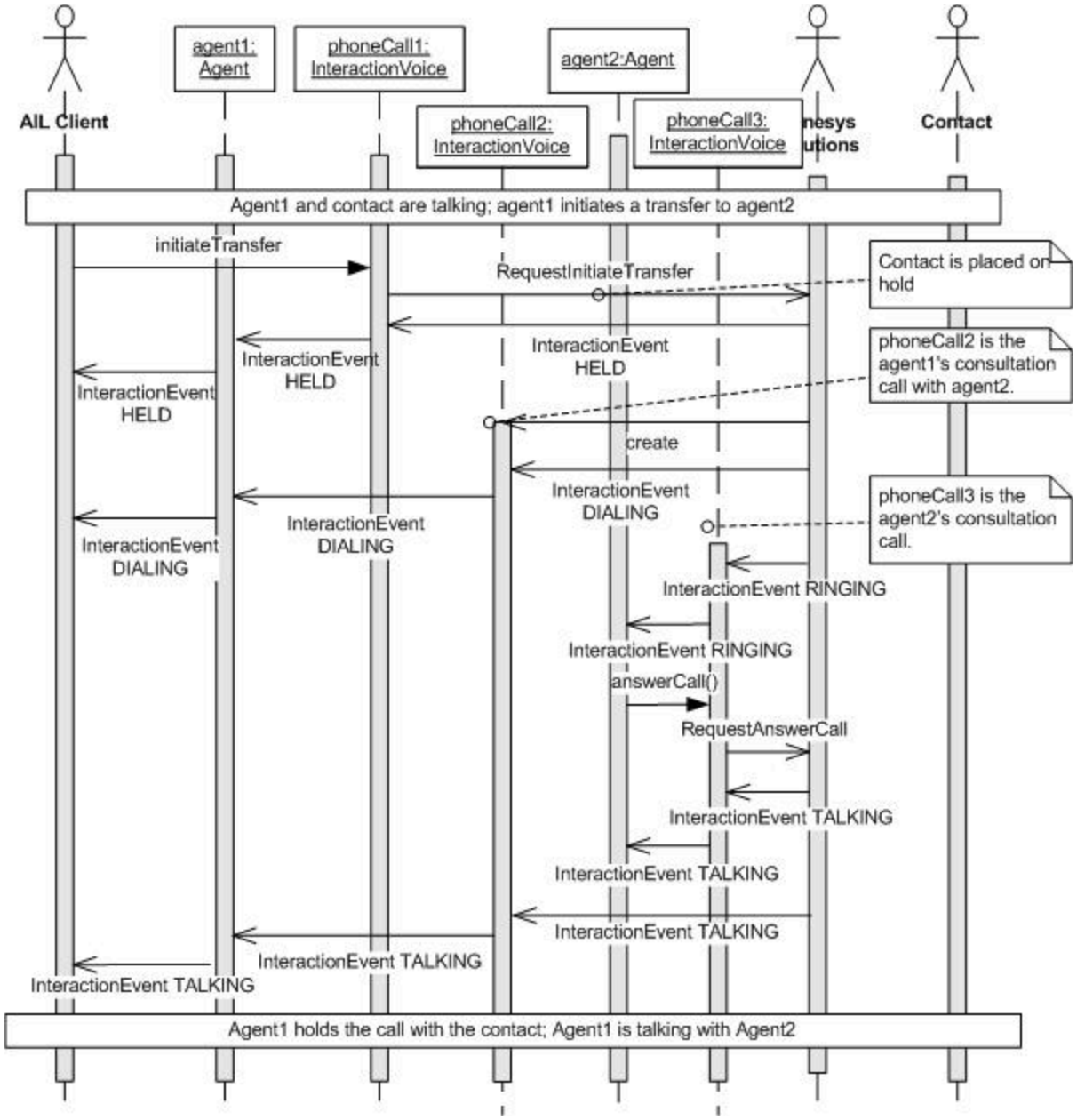


Call Conferencing, Completing

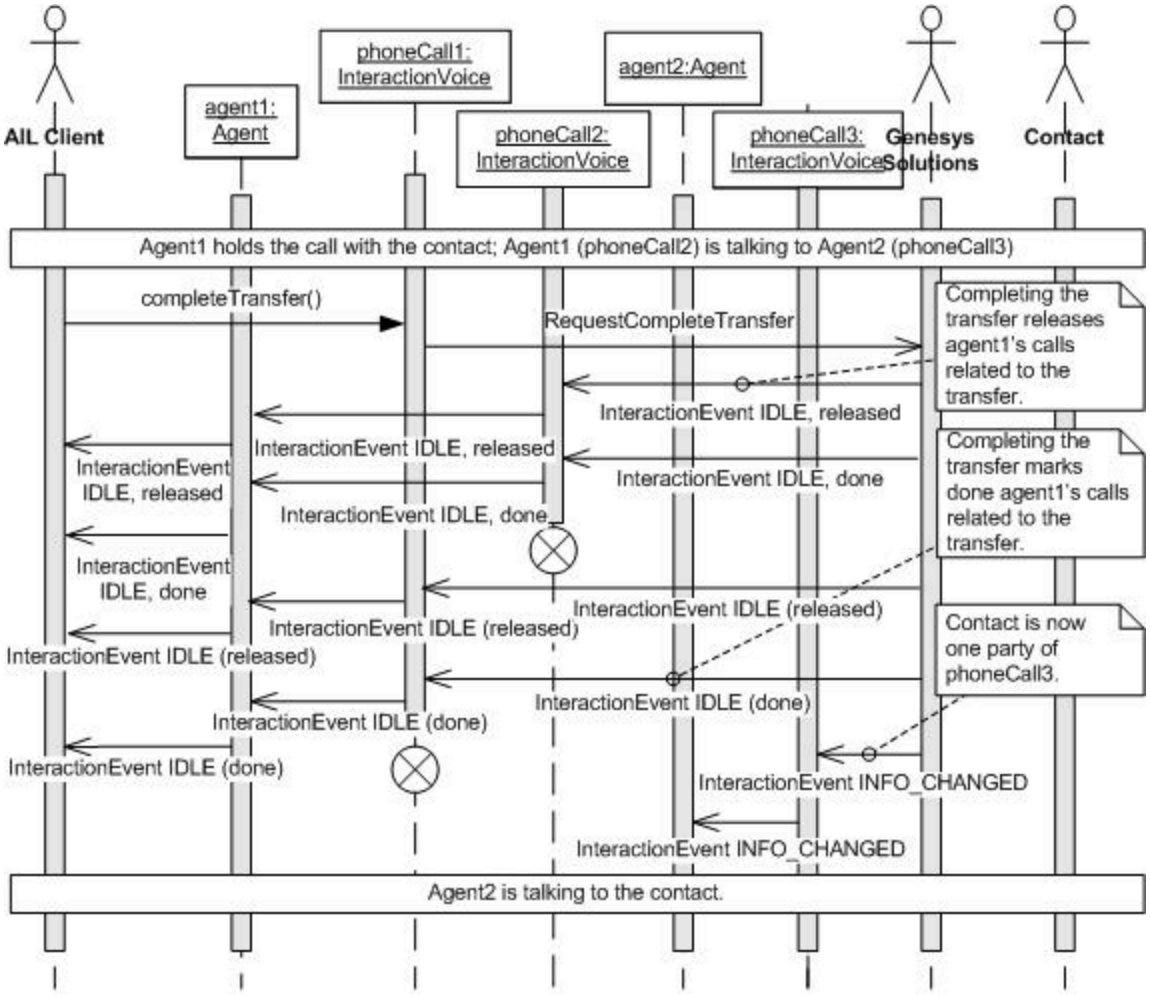
Transferring a Phone Call

Transferring a phone call follows the same steps as conferencing. Of course, all conference method calls must be replaced here by their transfer counterparts.
The difference lies in the fact that `agent1` does not resume its `InteractionVoice` `phoneCall1` at the end of the transfer. That is, instead of setting its first `InteractionVoice` to the status `TALKING` after the complete call, it sets it to `IDLE` and destroys it.

The event flow is presented in following two figures.



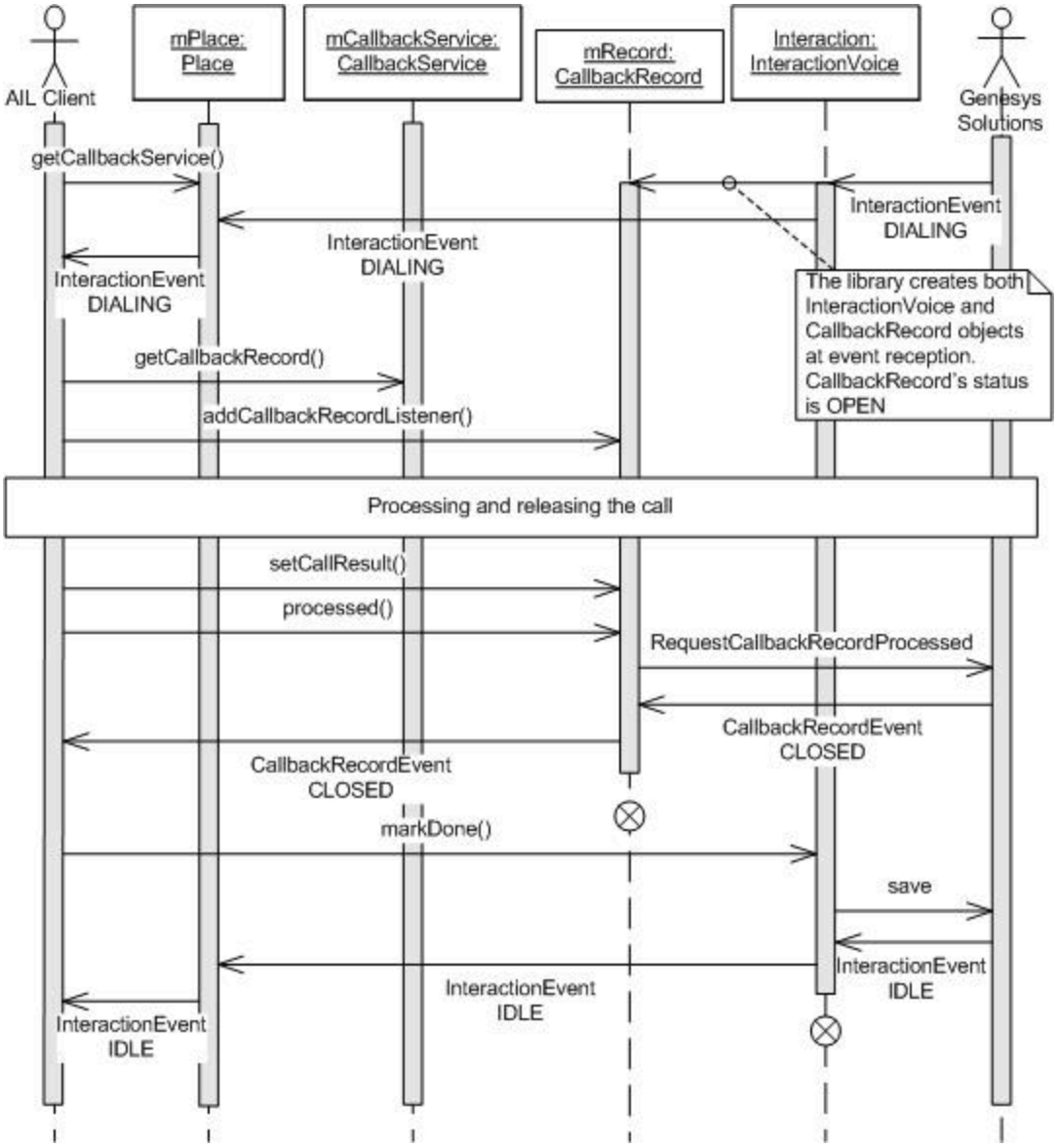
Transferring a Call, Initiating



Transferring a Call, Completing

Handling a Callback Phone Call

The following diagram shows the event flow for a callback record from a Callback Server in predictive mode.



Managing a Callback Record in a Predictive Callback Campaign