



# **Avaya Proactive Contact Agent API**

## **5.2**

Issue 1.0  
July 2018

© 2018, Avaya, Inc.  
All Rights Reserved.

#### **Notice**

While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

#### **Documentation disclaimer**

"Documentation" means information published by Avaya in varying mediums which may include product information, operating instructions and performance specifications that Avaya may generally make available to users of its products and Hosted Services.

Documentation does not include marketing materials. Avaya shall not be responsible for any modifications, additions, or deletions to the original published version of documentation unless such modifications, additions, or deletions were performed by Avaya. End User agrees to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation, to the extent made by End User.

#### **Link disclaimer**

Avaya is not responsible for the contents or reliability of any linked websites referenced within this site or documentation provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

#### **Warranty**

Avaya provides a limited warranty on Avaya hardware and software.

Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty language, as well as information regarding support for this product while under warranty is available to Avaya customers and other parties through the Avaya Support website:

<https://support.avaya.com/helpcenter/getGenericDetails?detailId=C20091120112456651010> under the link "Warranty & Product Lifecycle" or such successor site as designated by Avaya. Please note that if You acquired the product(s) from an authorized Avaya Channel Partner outside of the United States and Canada, the warranty is provided to You by said Avaya Channel Partner and not by Avaya.

"Hosted Service" means a hosted service subscription that You acquire from either Avaya or an authorized Avaya Channel Partner (as applicable) and which is described further in Hosted SAS or other service description documentation regarding the applicable hosted service. If You purchase a Hosted Service subscription, the foregoing limited warranty may not apply but You may be entitled to support services in connection with the Hosted Service as described further in your service description documents for the applicable Hosted Service. Contact Avaya or Avaya Channel Partner (as applicable) for more information.

#### **Hosted Service**

THE FOLLOWING APPLIES IF YOU PURCHASE A HOSTED SERVICE SUBSCRIPTION FROM AVAYA OR AN AVAYA CHANNEL PARTNER (AS APPLICABLE), THE TERMS OF USE FOR HOSTED SERVICES ARE AVAILABLE ON THE AVAYA WEBSITE,

[HTTPS://SUPPORT.AVAYA.COM/LICENSEINFO](https://support.avaya.com/licenseinfo)

UNDER THE LINK "Avaya Terms of Use for Hosted Services" OR SUCH SUCCESSOR SITE AS DESIGNATED BY AVAYA, AND ARE APPLICABLE TO ANYONE WHO ACCESSES OR USES THE HOSTED SERVICE. BY ACCESSING OR USING THE HOSTED SERVICE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE DOING SO (HEREINAFTER REFERRED TO INTERCHANGEABLY AS "YOU" AND "END USER"), AGREE TO THE TERMS OF USE. IF YOU ARE ACCEPTING THE TERMS OF USE ON BEHALF A COMPANY OR OTHER LEGAL ENTITY, YOU REPRESENT THAT YOU HAVE THE AUTHORITY TO BIND SUCH ENTITY TO THESE TERMS OF USE. IF YOU DO NOT HAVE SUCH AUTHORITY, OR IF YOU DO NOT WISH TO ACCEPT THESE TERMS OF USE, YOU MUST NOT ACCESS OR USE THE HOSTED SERVICE OR AUTHORIZE ANYONE TO ACCESS OR USE THE HOSTED SERVICE. YOUR USE OF THE HOSTED SERVICE SHALL BE LIMITED BY THE NUMBER AND TYPE OF LICENSES PURCHASED UNDER YOUR CONTRACT FOR THE HOSTED SERVICE, PROVIDED, HOWEVER, THAT FOR CERTAIN HOSTED SERVICES IF APPLICABLE, YOU MAY HAVE THE OPPORTUNITY TO USE FLEX LICENSES, WHICH WILL BE INVOICED ACCORDING TO ACTUAL USAGE ABOVE THE CONTRACT LICENSE LEVEL. CONTACT AVAYA OR AVAYA'S CHANNEL PARTNER FOR MORE INFORMATION ABOUT THE LICENSES FOR THE APPLICABLE HOSTED SERVICE, THE AVAILABILITY OF ANY FLEX LICENSES (IF APPLICABLE), PRICING AND BILLING INFORMATION AND OTHER IMPORTANT INFORMATION REGARDING THE HOSTED SERVICE.

#### **Licenses**

THE SOFTWARE LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE,

[HTTPS://SUPPORT.AVAYA.COM/LICENSEINFO](https://support.avaya.com/licenseinfo),

UNDER THE LINK "AVAYA SOFTWARE LICENSE TERMS (Avaya Products)" OR SUCH SUCCESSOR SITE AS DESIGNATED BY AVAYA, ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AVAYA CHANNEL PARTNER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AVAYA CHANNEL PARTNER.

UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA CHANNEL PARTNER; AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING OR SELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS "YOU" AND "END USER"), AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE AVAYA AFFILIATE ("AVAYA").

Avaya grants You a license within the scope of the license types described below, with the exception of Heritage Nortel Software, for which the scope of the license is detailed below. Where the order documentation does not expressly identify a license type, the applicable license will be a Designated System License. The applicable number of licenses and units of capacity for which the license is granted will be one (1), unless a different number of

licenses or units of capacity is specified in the documentation or other materials available to You. "Software" means computer programs in object code, provided by Avaya or an Avaya Channel Partner, whether as stand-alone products, pre-installed on hardware products, and any upgrades, updates, patches, bug fixes, or modified versions thereto. "Designated Processor" means a single stand-alone computing device. "Server" means a Designated Processor that hosts a software application to be accessed by multiple users.

"Instance" means a single copy of the Software executing at a particular time: (i) on one physical machine; or (ii) on one deployed software virtual machine ("VM") or similar deployment.

#### **License types**

**Designated System(s) License (DS).** End User may install and use each copy or an Instance of the Software only on a number of Designated Processors up to the number indicated in the order. Avaya may require the Designated Processor(s) to be identified in the order by type, serial number, feature key, Instance, location or other specific designation, or to be provided by End User to Avaya through electronic means established by Avaya specifically for this purpose.

**Concurrent User License (CU).** End User may install and use the Software on multiple Designated Processors or one or more Servers, so long as only the licensed number of Units are accessing and using the Software at any given time. A "Unit" means the unit on which Avaya, at its sole discretion, bases the pricing of its licenses and can be, without limitation, an agent, port or user, an e-mail or voice mail account in the name of a person or corporate function (e.g., webmaster or helpdesk), or a directory entry in the administrative database utilized by the Software that permits one user to interface with the Software. Units may be linked to a specific, identified Server or an Instance of the Software.

#### **Heritage Nortel Software**

"Heritage Nortel Software" means the software that was acquired by Avaya as part of its purchase of the Nortel Enterprise Solutions Business in December 2009. The Heritage Nortel Software is the software contained within the list of Heritage Nortel Products located at <https://support.avaya.com/LicenseInfo> under the link "Heritage Nortel Products" or such successor site as designated by Avaya. For Heritage Nortel Software, Avaya grants Customer a license to use Heritage Nortel Software provided hereunder solely to the extent of the authorized activation or authorized usage level, solely for the purpose specified in the Documentation, and solely as embedded in, for execution on, or for communication with Avaya equipment. Charges for Heritage Nortel Software may be based on extent of activation or use authorized as specified in an order or invoice.

#### **Copyright**

Except where expressly stated otherwise, no use should be made of materials on this site, the Documentation, Software, Hosted Service, or hardware provided by Avaya. All content on this site, the documentation, Hosted Service, and the product provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software unless expressly authorized by Avaya. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya can be a criminal, as well as a civil offense under the applicable law.

#### **Virtualization**

The following applies if the product is deployed on a virtual machine. Each product has its own ordering code and license types. Note that each Instance of a product must be separately licensed and ordered.

For example, if the end user customer or Avaya Channel Partner would like to install two Instances of the same type of products, then two products of that type must be ordered.

#### **Third Party Components**

"Third Party Components" mean certain software programs or portions thereof included in the Software or Hosted Service may contain software (including open source software) distributed under third party agreements ("Third Party Components"), which contain terms regarding the rights to use certain portions of the Software ("Third Party Terms"). As required, information regarding distributed Linux OS source code (for those products that have distributed Linux OS source code) and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply is available in the products, Documentation or on Avaya's website at: <https://support.avaya.com/Copyright> or such successor site as designated by Avaya. The open source software license terms provided as Third Party Terms are consistent with the license rights granted in these Software License Terms, and may contain additional rights benefiting You, such as modification and distribution of the open source software. The Third Party Terms shall take precedence over these Software License Terms, solely with respect to the applicable Third Party Components to the extent that these Software License Terms impose greater restrictions on You than the applicable Third Party Terms.

The following applies if the H.264 (AVC) codec is distributed with the product. THIS PRODUCT IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO (i) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (ii) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE [HTTP://WWW.MPEGLA.COM](http://www.mpegla.com).

#### **Service Provider**

THE FOLLOWING APPLIES TO AVAYA CHANNEL PARTNER'S HOSTING OF AVAYA PRODUCTS OR SERVICES. THE PRODUCT OR HOSTED SERVICE MAY USE THIRD PARTY COMPONENTS SUBJECT TO THIRD PARTY TERMS AND REQUIRE A SERVICE PROVIDER TO BE INDEPENDENTLY LICENSED DIRECTLY FROM THE THIRD PARTY SUPPLIER. AN AVAYA CHANNEL PARTNER'S HOSTING OF AVAYA PRODUCTS MUST BE AUTHORIZED IN WRITING BY AVAYA AND IF THOSE HOSTED PRODUCTS USE OR EMBED CERTAIN THIRD PARTY SOFTWARE, INCLUDING BUT NOT LIMITED TO MICROSOFT SOFTWARE OR CODECS, THE AVAYA CHANNEL PARTNER IS REQUIRED TO INDEPENDENTLY OBTAIN ANY APPLICABLE LICENSE AGREEMENTS, AT THE AVAYA CHANNEL PARTNER'S EXPENSE, DIRECTLY FROM THE APPLICABLE THIRD PARTY SUPPLIER.

WITH RESPECT TO CODECS, IF THE AVAYA CHANNEL PARTNER IS HOSTING ANY PRODUCTS THAT USE OR EMBED THE G.729 CODEC, H.264 CODEC, OR H.265 CODEC, THE AVAYA CHANNEL PARTNER ACKNOWLEDGES AND AGREES THE AVAYA CHANNEL PARTNER IS RESPONSIBLE FOR ANY AND ALL RELATED FEES AND/OR

ROYALTIES. THE G.729 CODEC IS LICENSED BY SIPRO LAB TELECOM INC. SEE [WWW.SIPRO.COM/CONTACT.HTML](http://WWW.SIPRO.COM/CONTACT.HTML). THE H.264 (AVC) CODEC IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO: (I) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (II) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION FOR H.264 (AVC) AND H.265 (HEVC) CODECS MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE [HTTP://WWW.MPEGLA.COM](http://WWW.MPEGLA.COM).

#### **Compliance with Laws**

Customer acknowledges and agrees that it is responsible for complying with any applicable laws and regulations, including, but not limited to laws and regulations related to call recording, data privacy, intellectual property, trade secret, fraud, and music performance rights, in the country or territory where the Avaya product is used.

#### **Preventing Toll Fraud**

"Toll Fraud" is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or is not working on your company's behalf). Be aware that there can be a risk of Toll Fraud associated with your system and that, if Toll Fraud occurs, it can result in substantial additional charges for your telecommunications services.

#### **Avaya Toll Fraud intervention**

If You suspect that You are being victimized by Toll Fraud and You need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Support website:

<https://support.avaya.com> or such successor site as designated by Avaya.

#### **Security Vulnerabilities**

Information about Avaya's security support policies can be found in the Security Policies and Support section of <https://support.avaya.com/security>.

Suspected Avaya product security vulnerabilities are handled per the Avaya Product Security Support Flow (<https://support.avaya.com/css/P8/documents/100161515>).

#### **Downloading Documentation**

For the most current versions of Documentation, see the Avaya Support website: <https://support.avaya.com>, or such successor site as designated by Avaya.

#### **Contact Avaya Support**

See the Avaya Support website: <https://support.avaya.com> for product or Hosted Service notices and articles, or to report a problem with your Avaya product or Hosted Service. For a list of support telephone numbers and contact addresses, go to the Avaya Support website: <https://support.avaya.com> (or such successor site as designated by Avaya), scroll to the bottom of the page, and select Contact Avaya Support.

#### **Trademarks**

The trademarks, logos and service marks ("Marks") displayed in this site, the Documentation, Hosted Service(s), and product(s) provided by Avaya are the registered or unregistered Marks of Avaya, its affiliates, or other third parties. Users are not permitted to use such Marks without prior written consent from Avaya or such third party which may own the Mark. Nothing contained in this site, the Documentation, Hosted Service(s) and product(s)

should be construed as granting, by implication, estoppel, or otherwise, any license or right in and to the Marks without the express written permission of Avaya or the applicable third party. Avaya is a registered trademark of Avaya Inc. All non-Avaya trademarks are the property of their respective owners. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

# Contents

Preface .....	2
Purpose.....	2
Audience .....	2
Related documents .....	2
Availability .....	3
Overview .....	4
Avaya Proactive Contact System Functions.....	4
Multiple Dialers .....	4
Pods .....	5
Calling Lists .....	5
Agent Blending .....	6
Security Overview .....	8
Communication Encryption.....	8
Identity Authentication .....	8
Certificates Generation Signing and Maintenance .....	8
Secured Agent.....	9
Certificates .....	9
Agent Tasks Overview .....	9
Agent types .....	11
Log In to Proactive Contact .....	11
Join jobs .....	12
Handle Calls .....	12
Wrap Up Calls .....	14
Log out of Proactive Contact .....	15
Understanding the Proactive Contact Agent API.....	16
Terminology.....	16
Agent Applications.....	17
Agent Binary.....	20
Agent API Messages .....	20
Agent Application Initiated Messages.....	21
Agent Binary Initiated Messages .....	21
Message Formats.....	21
Agent States.....	25
Outbound Job Commands (by agent state).....	28
Outbound Job Message Scenarios .....	29

Moagent32.log .....	34
Sample Agent Application Session .....	34
Sample Managed Dialing Job .....	40
Sample Unit Work List Job .....	45
Sample Agent Owned Recall .....	47
Using Proactive Contact Agent API DLL .....	52
Developer requirements .....	52
The Proactive Contact Agent API and COM .....	53
Work with the Moagent32.dll .....	53
Connect to the Moagent32.dll .....	54
Use the interface methods .....	55
Handle notification events from Proactive Contact .....	57
Set default properties .....	59
Code samples .....	59
Use Moagent32.dll dialog boxes .....	61
The Moagent32 Interface .....	66
Method Syntax .....	66
Moagent32.ini Settings .....	67
[Logon] .....	67
[Security] .....	67
[Work_class] .....	68
[Dialog_box_text] .....	68
[Agent_state] .....	68
[Server_return_codes] .....	69
[Winsockerrors] .....	69
[development_platform] .....	69
Commands and Notification Events .....	70
AGTAdjustHeadset .....	70
AGTAORNotify .....	72
AGTAttachJob .....	73
AGTAutoReleaseLine .....	75
AGTAureIToReady .....	76
AGTAvailWork .....	78
AGTCallNotify .....	81
AGTCancelSearch .....	85
AGTClearDataSet .....	86
AGTConnHeadset .....	88
AGTDetachJob .....	90
AGTDialDigit .....	91
AGTDialerMode .....	93

AGTDisconnHeadset .....	94
AGTDoNotCall .....	96
AGTDumpData.....	97
AGTEchoOff.....	101
AGTEchoOn.....	103
AGTFindFirstRecord .....	104
AGTFindNextRecord .....	106
AGTFinishedItem .....	108
AGTFreeHeadset .....	110
AGTGetHeadsetVol .....	112
AGTGetScreen.....	113
AGTHangupCall .....	115
AGTHeadsetConnBroken.....	116
AGTHoldCall .....	118
AGTHookflashLine .....	119
AGTlicbAbort.....	121
AGTlicbFeNotif.....	122
AGTlicbOffline.....	124
AGTlicbOnline .....	125
AGTJobEnd.....	126
AGTJobTransLink .....	128
AGTJobTransRequest .....	129
AGTListCallbackFmt .....	131
AGTListCallFields .....	133
AGTListCallLists.....	135
AGTListDataFields .....	136
AGTListJobs.....	139
AGTListKeys .....	140
AGTListScreens .....	142
AGTListState.....	143
AGTListUnits .....	146
AGTLogloStart .....	147
AGTLogloStop .....	153
AGTLogoff .....	154
AGTLogoffAcid.....	155
AGTLogon.....	157
AGTLogonAcid.....	159
AGTManagedCall.....	161
AGTManCallAnswered.....	164
AGTManualCall.....	164

AGTManualPhone .....	166
AGTMoFlashBlind .....	168
AGTMoFlashSupv .....	172
AGTNoFurtherWork .....	176
AGTPreviewRecord .....	178
AGTReadField .....	181
AGTReadyNextItem .....	183
AGTReceiveMessage .....	185
AGTJobMode .....	186
AGTReleaseLine .....	188
AGTReserveHeadset .....	190
AGTSendMessage .....	192
AGTSetCallback .....	193
AGTSetDataField .....	198
AGTSetNotifyKeyField .....	200
AGTSetPassword .....	202
AGTSetTenant .....	203
AGTSetUnit .....	205
AGTSetWorkClass .....	208
AGTSTART .....	210
AGTSystemError .....	211
AGTTransferCall .....	213
AGTUnholdCall .....	215
AGTUnitEnd .....	217
AGTUpdateField .....	218
AGTXferCustHangup .....	220
AGTXferTrunkHangup .....	221
Appendix A: Completion codes .....	223
Appendix B: Agent API Message Format Quick Reference .....	227
Command Message Formats .....	227
Response Message Formats .....	229
Data Message Formats .....	229
Notification Event Message Formats .....	231
Appendix C: System Messages .....	234
Error Messages .....	234
Data Messages .....	274
Pending Messages .....	277
Appendix D: Moagent32.dll Interfaces .....	280
IServerStartup Interface .....	280
IConfigure Interface .....	280



Properties .....	281
Events .....	282
Methods.....	282
IMoagent Interface .....	282
Properties .....	282
Event .....	282
Methods.....	283
Appendix E: Data Parameters.....	286
Appendix F: Use the Agent DDE Interface .....	290
Events and Methods .....	290
CallType .....	290
CallFields.....	290
CallNotify .....	290
PreviewFields .....	291
PreviewNotify .....	291
ManagedCall .....	291
FieldChanged .....	291
PushField .....	291
Command Line Flags .....	291
Appendix G: Avaya Proactive Contact 5.1.2/5.1.3 Agent API clients interacting with PC 5.2 Dialer .....	293
Index .....	294

# Preface

This section contains the following topics:

[Purpose](#)

[Audience](#)

[Reason for reissue](#)

[Related documents](#)[Availability](#)

## Purpose

The purpose of this guide is to provide detailed information about Avaya Proactive Contact.

Note: The latest and most accurate version of Agent API SDK Guide can be downloaded from the Avaya support site at <http://support.avaya.com>.

## Audience

This guide is intended primarily for those who use Proactive Contact. You should use this guide as an information source for changing the configuration of your Agent API.

Reason for issuing The Agent API is updated with the following information:

- Added API AGTListTenants and AGTSetTenant for Multi Tenancy.
- Added API AGTAutorelToReady
- Added support to specify the tenant name with agent name in AGTLogon command and added return error codes for tenant.
- Added return error code E58021 in AGTAttachJob.

The Agent API is updated with the following on 5.1.1 version:

- Added return error code E28917 in AGTSystemError

The Agent API is updated with the following on 5.1.3 version:

- Added return error code E70000 and E28644 in AGTListCallFields
- Added support for TLS v1.2 in 5.2 version

**Note:** You need to upgrade the custom client using the API to take advantage of the new features or modifications in the latest Agent API.

## Related documents

The Avaya Proactive Contact documentation set consists of:

*Overview of Avaya Proactive Contact*

*Administering Avaya Proactive Contact*

*Using Avaya Proactive Contact Supervisor*

*Using Avaya Proactive Contact Agent*

*Software Developer's Kit (SDK ) for Avaya Proactive Contact*

*Planning and Prerequisites for Avaya Proactive Contact*

*Avaya Proactive Contact Safety and Regulatory Information*

## **Availability**

The latest version of this document is available on the Avaya online support Web site:

<http://support.avaya.com>

## Overview

The Agent API system consists of software, hardware, and network components. The system is comprised of the system cabinet, supervisor workstation, agent workstations, printer, and modem.

You can use the UNIX-based interface to the system from either the administrator workstation or any personal computer using Telnet. To access the Administrator menus, you must have an account set up with administrator privileges. Using the Supervisor menus, you can perform many of the same tasks available in the Avaya Proactive Contact Supervisor applications if you have an account set up with supervisor privileges.

The system works with your call center's equipment and operations to perform call center tasks. Your installation can include more than one dialer.

This section contains the following topics:

- [Avaya Proactive Contact System Functions](#)
- [Agent Tasks Overview](#)
- [Understanding the Proactive Contact](#)

## Avaya Proactive Contact System Functions

The following list describes the main functions of the system:

- Receives customer records from the call center's host computer.
- Selects and sorts customer records based on your call center's business goals.
- Allows agents to update customer information on an agent screen or on the host, depending on your configuration.
- Passes only specific call types to agents.
- Adjusts the calling pace to meet the call center's requirements.
- Monitors ACD inbound traffic and predicts when to acquire and release ACD agents for outbound calling on Agent API with Agent Blending.
- Supports outbound, inbound, and blend jobs.
- Generates a variety of reports, including job, agent, and system reports.
- Uploads record information to the host (optional).

## Multiple Dialers

Your Agent API system can include multiple dialers. You can connect up to ten dialers through a Mid-Tier structure.

An Agent API system that is connected through a Mid-Tier structure is a pod.

Your Agent API system can also have a distributed architecture. The system can use dialers in the following architecture:

- Multiple stand alone dialers
- One or more pods of dialers
- Multiple stand alone dialers and multiple pods of dialers

## **Pods**

A multiple dialer office environment that uses a pod increases your company's outreach capacity. A pod allows you to manage large-scale outreach programs from one administration and Supervisor interface.

A pod provides additional benefits including the following features:

- Calling lists
- Jobs
- Phone strategies
- Record selections
- Logins

From one Supervisor application, you can run one job on multiple dialers and monitor the calling activities on each dialer.

## **Calling Lists**

A calling list is a file that contains customer records. Agent API uses two types of calling lists, one for outbound calling and one for inbound calling on Intelligent Call Blending systems.

The host system creates the download file of customer records for the outbound calling list. The download file contains the records and fields you defined as necessary to your outbound call activities.

Proactive Contact processes the host file and prepares it for the calling activities. When the calling activities end, the system prepares the calling list to be uploaded to the host.

## **Process Calling Lists**

After the host downloads the customer records, Agent API completes the following tasks to create a calling list:

- Checks for and flags duplicate records and invalid phone numbers.
- Identifies and marks records that have been on the system more than a specified number of days.
- Recalls the name of the last agent to speak to the customer.
- Stores the result of the last call attempt as recorded by the agent.
- Verifies the following statistics:
  - Name of the last agent to speak with the customer
  - Date and time of the last call attempt

- Result of the last call attempt as recorded by the agent on the system
- Number of days the record has been on the system
- Record status

After calling activities, at a scheduled time, Agent API completes the following tasks to upload the file to the host:

- Converts the customer records in a specific calling list to format specified for your host computer.
- Creates an upload file.

The host then updates your customer database with the data in the converted calling list.

## **Environment**

The calling list environment is responsible for these activities:

- Create the files required to convert host computer data to the Agent API calling list format.
- Prepare the calling list for the calling activities.
- Prepare the calling list for extracting data to send back to the host after calls have been made.

## **Agent Blending**

Agent Blending is a tool that integrates outbound calling activities on your Agent API system with inbound calling activities on your ACD. The Agent Blending tool allows you to manage the ACD domains and domain groups. A domain is an ACD call queue. Every domain is a member of a domain group.

Agent API provides two types of Agent Blending: Predictive Agent Blending and Proactive Agent Blending.

Both types of Agent Blending systems use a pool of ACD blend agents for outbound calling. The ACD agents log in to the dialer and the ACD. Agent Blending monitors the activity on the ACD to determine when to move agents between inbound and outbound calling activities.

The dialer acquires the pooled agents for outbound calling when the inbound calling activity decreases. The dialer releases the pooled agents to inbound calling when the inbound calling activity increases. The movement between inbound and outbound calling keeps the ACD blend agents busy and the ACD service level within your prescribed limits.

Use Predictive Agent Blending if your priority is servicing your inbound customers and your inbound volume is fairly high.

Use Predictive Agent Blending if your call center has the following amount of work:

- Moderate to heavy inbound traffic
- More than 25 agents in an inbound pool

## Predictive Agent Blending

Use Predictive Agent Blending if your priority is servicing your inbound customers and your inbound volume is fairly high.

Predictive Agent Blending focuses on the inbound mission. Predictive Agent Blending uses events from the ACD to forecast call volume and determine when to move ACD agents between inbound and outbound calling. The dialer predicts when too many agents receive inbound calls. The dialer then acquires agents from the ACD to handle outbound calls until the inbound volume increases.

Agent API acquires agents for outbound calls when either the settings for the **Average Speed to Answer** or **Service Level** domain groups are above the desired value.

To configure Predictive Agent Blending, set up an Average Speed to Answer or a Service Level domain group that contains one or more acquire domains and at least one inbound domain.

### Average Speed to Answer (ASA)

This domain group type uses the target ASA field (MAAS) to calculate when to acquire and release agents. The dialer:

- Acquires agents for outbound calls when the average speed to answer for all inbound domains in the group is less than or equal to the targeted value.
- Releases agents when the value rises above the target.

### Service Level (SL)

This domain group type uses the **Service Criterion** (SC, seconds), **Desired Service Level** (DSL, %), and **Abatement Service Level** (ASL, %) fields for calculating when to acquire and release agents.

- The dialer acquires agents for outbound calls when the percentage of inbound calls answered within the Service Criterion is greater than or equal to the Desired Service Level percentage.
- The dialer no longer acquires agents when the actual service level reaches the Abatement Service Level value.
- The dialer releases agents to inbound when the service level falls below the desired value.

The actual service level is calculated using all inbound domains in the group.

## Proactive Agent Blending

Use Proactive Agent Blending if your focus is on outbound calling, but you need to service a low volume of inbound customers.

Proactive Agent Blending focuses on outbound calls and releases agents to inbound only when an inbound call enters a monitored queue on the ACD.

When an ACD agent logs in, the dialer immediately acquires the agent for outbound calling. When an inbound call arrives in the ACD queue, the dialer releases the agent

to handle the call. If inbound calls continue to arrive, the dialer continues to release agents. When the queue is empty, the dialer acquires agents for outbound calls.

---

For each OB\_ONLY domain group, you configure the number of queued calls before agents release to inbound.

## Security Overview

The Agent application is secured that offers the following benefits:

- Communicates using Secure Socket Layer (SSL/TLS).
- Data Transmission is encrypted.
- Certificates are used for client and server authentication.
- Security Settings are saved in Moagent32.ini file.

## Communication Encryption

Proactive Contact 4.0 and later adopts a secure protocol called SSL for all its communications. SSL fulfills requirements that make it acceptable for use in the transmission of even the most sensitive of transactions, such as credit card information, medical records, legal documents, and e-commerce applications.

SSL encrypts data before it is sent. The receiver decrypts it after it is delivered. The encryption mechanisms are only known to the sender and the receiver.

## Identity Authentication

In addition to operating system, password-based authentication, Avaya Proactive Contact 5.1 also validates incoming client certificates. Servers (Agent/ Enserver /List Server) receives client's certificates during the SSL handshake. Based on the server configuration, it validates the client if the certificate is valid and signed by trusted Certificate Authority (CA).

## Certificates Generation Signing and Maintenance

Currently, OpenSSL CA and EJBCA are the types of CAs evaluated . EJBCA is the Avaya Proactive Contact 5.1 certificate management tool. EJBCA is a fully functional CA built in Java and based on J2EE standards.

OpenSSL CA is an open source CA distributed from <http://openssl.org>. It is a minimal CA application. It signs certificate requests in a variety of forms and generate CRLs. It also maintains a text database of issued certificates and their status.

Avaya Proactive Contact 5.1 systems have a set of default certificates for all the services and internal clients. The Avaya Proactive Contact 5.1 Agent API provides default certificates for external clients.

The certificates can be generated in several ways:

- A single certificate is shared by all the services.



- A single certificate is shared by all the internal clients. A certificate is generated for each service or internal client. All these options are designated in a configuration file.

Also in the configuration file, it provides information on whether both servers and clients have certificates, or only servers have certificates.

## Secured Agent

The Agent application is secured that offers the following benefits:

- Communicates using Secure Socket Layer (SSL/TLS).
- Encrypts data transmission.
- Uses certificates for client and server authentication.
- Saves security settings in the PDSAgent.ini file.

## Certificates

A list of certificates that are required by Proactive Contact Agent is as follows:

- Certificate Key File: This file stores the agent client certificate key.
- Certificate File: This file stores the agent client certificate.
- CA Certificate: This certificate is stored in Microsoft Certificate Store and authenticates the server certificate.

The user must add the CA certificate in the trusted CA store on windows where the custom agent application is running. Client certificate signed by CA is supplied along with the Agent API. Agent API uses this certificate internally for connecting to the Server.

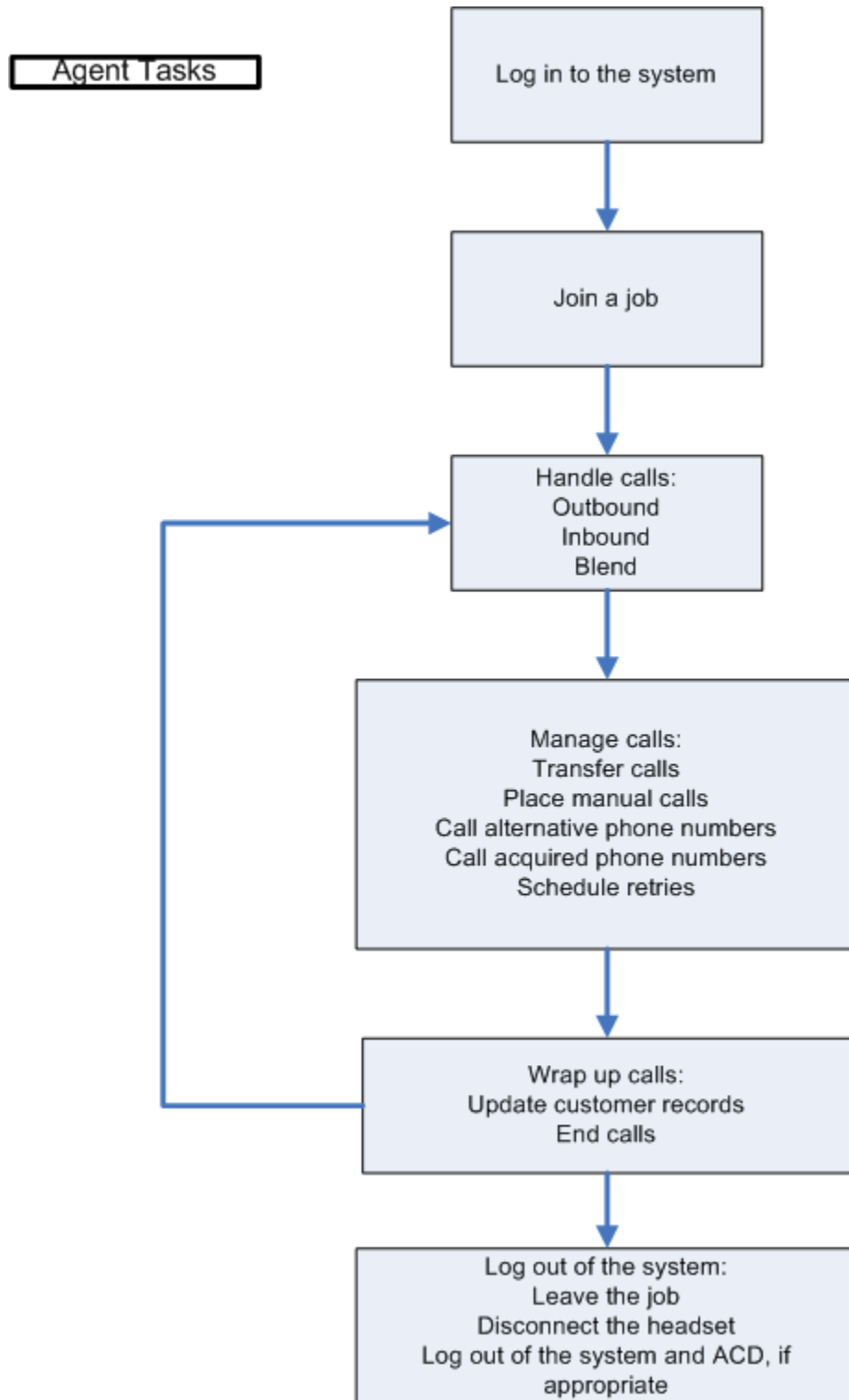
## Agent Tasks Overview

Call center agents talk with customers and update customer records. To develop agent applications, you need to understand agent tasks. This section provides overview information to help you understand these tasks. For more thorough documentation on agent tasks, see the online Help that comes with the Proactive Contact Agent software application.

This section contains the following topics:

- [Agent types](#)
- [Log In to Proactive Contact](#)
- [Join jobs](#)
- [Handle Calls](#)
- [Wrap Up Calls](#)
- [Log out of Proactive Contact](#)

The following diagram illustrates the discussion. Both focus on the activities agents perform without an agent application. [Understanding the Proactive Contact](#) covers how to automate some procedures.



## Agent types

Agents work on specific types of jobs by logging in to the system as a specific type of agent. The agent type determines which features are available and which jobs agents can handle.

Agent type	Description
Outbound Agent	Outbound and blend jobs Agents handle calls Proactive Contact places.
Inbound Agent	Inbound and blend jobs Agents handle calls received from customers.
Blend Agent	Blend jobs Agents handle outbound and inbound calls.
Managed Agent	Managed Dialing jobs Agents handle only outbound calls.
Person to Person Agent	Any job Agents handle outbound calls when outbound agents aren't available. They typically perform other work between outbound calls.
ACD Agent	Agent Blending ACD agents log in to both Proactive Contact and the ACD. They handle outbound calls from Proactive Contact when not handling inbound calls from the ACD. ACD agents using Agent API are ACD agents dedicated to outbound calling. These agents log in to the ACD and belong to the "ACD Outbound Only" domain group.

## Log In to Proactive Contact

The login process follows these steps:

1. Agents enter a user name or user identification number (ID) and a password to log in to Proactive Contact.

---

On Proactive Contact, the login process distinguishes between uppercase and lowercase letters. For example, MARY, Mary, and mary are different user names.

2. The agent enters an identification number (ID) that tells Proactive Contact how to make the audio connection to the agent through the agent headset or workstation ID.

3. Proactive Contact confirms that the ID is valid and then displays the agent interface. The default interface is the Proactive Contact Agent.
4. Proactive Contact establishes a voice connection to the agent headset. The agent hears a welcome message when Proactive Contact makes the connection.

---

If the Agent is an ACD agent, the agent also logs on to the ACD.

## Join jobs

After establishing workstation and voice connections, the agent joins an active job. If the agent type does not match the job type, such as an inbound agent for an outbound job, Proactive Contact will not accept the job selection or the agent cannot even see the other jobs that they are not allowed to log in to.

If the job uses unit work lists, the agent selects a value for the unit ID. An example of a unit work list is that Spanish-speaking agents log into a Spanish unit work list in a job.

Finally, agents tell Proactive Contact that they are available to handle calls.

## Handle Calls

Proactive Contact distinguishes between calls answered by a person and calls answered by an electronic device. Proactive Contact refers to calls answered by a person as Live Calls. Calls answered by an electronic device are Autovoice calls.

The supervisor configures the job to determine which call types Proactive Contact passes to the agents. Supervisors may decide to pass Autovoice calls to agents so that agents can leave messages on the device, or they might decide to screen out these calls. The supervisor also decides whether or not to place all calls in a wait queue when no agents are available.

When Proactive Contact connects calls to the agent, it uses three types of signals to announce calls.

- A ziptone sounds in the headset. On some systems, there is a difference between the tones for inbound and outbound calls.
- The sound level in the headset changes.
- Proactive Contact displays customer record information for an outbound call or a data entry screen for an inbound call.

On some systems, the agents see a special message when customers have been on hold. The message either tells the agent how long the customer has been on hold, or it displays text for the agent to read to the customer. For example, the text might be "Sorry you had to hold."

## Outbound Calling

Proactive Contact connects the outbound call to a customer to a blend or an outbound agent, and plays a tone in the agent's headset to announce each call. Proactive Contact displays the customer record on the agent's screen. The agent talks to the customer and updates the customer record.

## **Managed Outbound Dialing Calls**

For Proactive Contact, the term “Managed” refers to the fact that agents have a preview time prior to the call when they can review the customer’s information.

Proactive Contact connects the outbound calls to a Managed agent logged in to a Managed Dialing job.

The Avaya supervisor sets up a record preview period for Managed Dialing jobs. During this period, the agent can look at the customer record and decide if Proactive Contact should call the customer (although some supervisors disable this functionality so that agents are required to handle the calls).

If the agent waits until the preview period expires, Proactive Contact places the call automatically.

If the agent wants to place a call to the customer before the preview period ends, the agent tells the system to call the phone number immediately (through the Place Managed Call command).

If the agent decides not to call the customer, the agent executes the Cancel Managed Call command and Proactive Contact cancels the call, releases the line, releases the record, and sends the agent a new call. Cancelling the call is only valid before the preview period expires and it is sometimes disabled by the supervisor.

## **Inbound Calling**

Inbound or blend agents can join inbound jobs on Proactive Contact with Intelligent Call Blending to handle inbound calls. After the agent joins a job, Proactive Contact connects an inbound customer call to an agent after displaying an inbound call screen. This screen contains one or more blank fields for gathering customer information.

The agent talks to the customer and enters information on the screen. In some call centers, the agent can add customer information to Proactive Contact or the host system.

If the agent is updating customer information on the host, the agent sends the request for the customer record to the host. The host system software displays and updates the host records. This is not part of Proactive Contact.

## **Blend Calling**

Agents can join blend jobs on Proactive Contact with Intelligent Call Blending to take both outbound and inbound calls through Proactive Contact.

During a blend job, Proactive Contact places outbound calls and passes the calls to agents.

Most call centers receive inbound calls through their ACD. These call centers usually have some agents who are dedicated to handling inbound calls on the ACD. When there is an excess of inbound calls, the ACD directs calls to Proactive Contact blend agents. When there are no inbound calls overflowing from the ACD, Proactive Contact connects outbound calls to blend agents.

On systems without an ACD, Proactive Contact passes inbound calls directly to the blend agents.

## **Transferring Calls**

Agents can transfer inbound or outbound calls to another telephone number, a supervisor, or an extension. On Proactive Contact with Native Voice and Data Transfer, an agent can transfer both the call and customer record. Agents can supervise the transfer.

During an unsupervised call transfer, the agent can use a function key or a command in the Agent application to disconnect from the customer before the transfer party answers the phone, or the agent can stay on the line and announce the transfer.

During a supervised call transfer, the agent can disconnect the call transfer (manual hang up) and maintain contact with the customer. This is useful when a call transfer might reach a wrong number, busy signal, answering machine, or an error message.

## **Placing Manual Calls**

Agents can initiate and cancel calls from their workstations without releasing control of a telephone line. This lets the agent place calls to phone numbers received during a conversation or to a phone number displayed on the agent's screen.

## **Calling Alternate Phone Numbers**

In the UNIX-based agent interface, the agent screen displays fields that contain telephone numbers. The agent selects a customer telephone number and tells Proactive Contact to place the call. The system calls the phone number.

## **Calling Acquired Phone Numbers**

The agent might tell Proactive Contact to dial a phone number acquired during a conversation with a customer. The agent may dial this number using the Numbers text box in their Agent application. They type the phone number in the box and then select Place Manual Call.

## **Scheduling Recalls**

Some Proactive Contact configurations allow the agent to schedule recalls setting the phone number, and the time and date to call the customer back. On systems with agent-owned recall, agents can specify that they own the recall. When Proactive Contact places an agent-owned recall, it attempts to locate the agent who established the recall from the list of agents logged in to the system. If the agent is not logged in, Proactive Contact postpones the call.

## **Wrap Up Calls**

When the conversation ends, the agent might be finished with the customer's record and ready for another call, or the agent might need additional time to finish customer record updates.

## **Updating Customer Records**

Agents update the customer records on their screens and send the updated information to Proactive Contact or the host.

## Ending Calls

If the agent is ready for another call, the agent assigns a call completion code that identifies the conversation's outcome. Proactive Contact releases the telephone line, saves the customer record, releases the record, and records that the agent is available.

If the agent needs additional update time, the agent releases the telephone line and then finishes updating the customer record.

When the agent is ready to take another call, the agent assigns a call completion code that identifies the conversation's outcome. Proactive Contact saves the record update and releases the record. It then records that the agent is available for another call.

## Log out of Proactive Contact

Logging out has three steps: leaving a job, disconnecting the agent's headset, and logging out of Proactive Contact.

### Leave a job

Agents typically leave a job or go offline for the following reasons:

- To transfer to another job.
- To stop handling calls temporarily during the day, such as to take a break or attend a meeting.
- To log out at the end of the day.

To leave a job, the agent sends Proactive Contact an offline request. If the system was about to pass a call to the agent when it receives the offline request, it completes the connection to the agent.

Proactive Contact stops placing calls when all agents leave a job. However, if there are calls waiting on hold, the last agent remaining on the job receives calls until the call queue is empty.

### Disconnect Headsets

When the agent stops work for the day, Proactive Contact disconnects from the agent's headset and clears the memory buffer reserved for that headset identification. On most Proactive Contact agent interfaces, the logout process automatically disconnects the headset.

### Log Out

On the Proactive Contact UNIX-based Agent Menu, the agent selects logout. Proactive Contact logs the agent out of the system and returns the workstation to the Proactive Contact login prompt.

- If the agent is an ACD agent working on an Agent Blending system, the agent must log out of the ACD before logging out of Proactive Contact.
- Note that if the workstation also connects to a host system, the agent might need to log out the host system after logging out of Proactive Contact.

On the Proactive Contact UNIX-based Agent Menu, agents use the Quit function key only when they are unable to leave a job and log out using standard procedures. The

Quit key displays a screen message that prompts the agent to confirm that an emergency exit is necessary. Pressing the Quit key instructs Proactive Contact to terminate the calling session by disconnecting any call in progress and immediately logging the agent out of the system.

## Understanding the Proactive Contact Agent API

The Proactive Contact Agent API (Agent API) lets you build a graphical user interface (GUI) for Proactive Contact agents. This documentation refers to that GUI as the agent application.

The agent binary is the core of the Agent API. When using the Agent API, the agent binary is the server process running on the Proactive Contact server that enables an API for custom applications.

The agent application can run on any workstation that can establish a Secure Socket Layer (SSL) connection to Proactive Contact.

This section contains the following topics:

- [Terminology](#)
- [Agent Applications](#)
- [Agent Binary](#)
- [Agent API Messages](#)
- [Agent Application Initiated Messages](#)
- [Agent B](#)
- [Message F](#)
- [Agent S](#)
- [Outbound Job Commands \(by a](#)
- [Outbound J](#)
- [Moagent32.log](#)
- [Sample A](#)
- [Sample M](#)
- [Sample U](#)
- [Sample A](#)

### Terminology

The following definitions explain how terms are used in this guide.

Term	Definition
------	------------



Term	Definition
Agent application	The application Proactive Contact agents use to work with customers. The agent application runs on an agent's workstation.
Agent binary	The agent process runs on Proactive Contact and provides command driven access. Some Proactive Contact documentation might refer to the agent process as the Proactive Contact process.
Proactive Contact server	The network server running the Proactive Contact application.

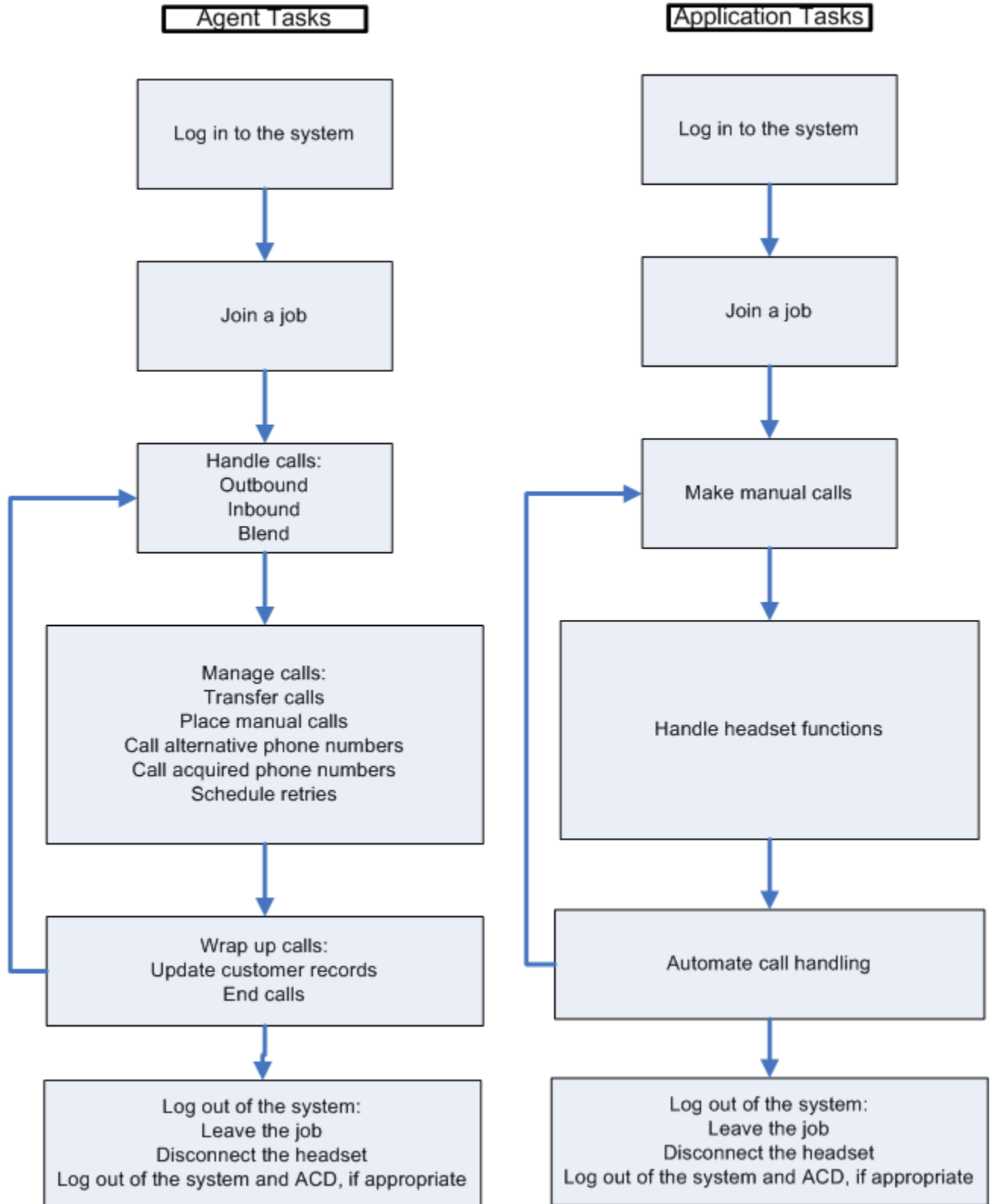
## Agent Applications

The agent application is the application agents logged in to Proactive Contact use to work with a customer. Agent API gives you the tools to create agent applications that automate many of the tasks that agents perform. To understand typical agent tasks, see [Agent Tasks Overview](#).

Agent applications using the Agent API can provide buttons, menus, or other controls available in your development tools to simplify and speed up agent tasks. In addition to the Agent API, Windows developers have access to the optional dialog boxes provided by the Moagent32.dll. Agents use dialog boxes to enter information that the agent binary will pass to Proactive Contact.

This section illustrates and describes some tasks the Agent API can perform. It also references related Moagent32.dll functions as examples.

The following diagram illustrates agent tasks that the agent application can automate.



## **Log in to Proactive Contact**

Agent API can automate the login process for each workstation, requiring the agent to enter a user name, password, extension, and agent type. Moagent32.dll provides a standard login dialog box as an option.

## **Join a Job**

Agent API automates the tasks associated with selecting and joining a job. The agent application can collect the information it needs about a job and send commands to the agent binary without the agent's intervention. The Moagent32.dll provides standard dialog boxes to choose between agent types, jobs, and job parameters (such as unit ID selection for a Unit Work List job).

## **Release Lines and Transfer Calls**

Agent API provides optional dialog boxes which enable an agent to release telephone lines and transfer calls.

## **Automate Call Handling**

Agent API releases customer records as specified in your application and enables agents to select messages to play (AGTReleaseLine).

## **Handle Headset Functions**

Agent API provides commands to allow an agent application to adjust headset volumes on systems using OLIC cards. Moagent32.dll provides a dialog box for headset volume adjustment on systems using OLIC cards.

## **Log out of Proactive Contact**

An agent application developed with the Agent API can check ACD status and provide separate logout processes for ACD agents and Proactive Contact agents.

Moagent32.dll provides a standard logout process that issues the appropriate commands.

If the workstation also connects to a host system, the agent might need to log out of the host system connection after logging out of Proactive Contact. The Agent API does not provide for logging out of a host system connection.

## **Exit Proactive Contact**

Agent API does not support agent-initiated emergency exits from Proactive Contact. If the system connected to the agent application malfunctions, the agent binary automatically terminates all agent application sessions.

## **Testing Applications**

You need access to Proactive Contact to test your agent application. We recommend that you work closely with the Proactive Contact supervisor at the call center. An agent workstation in a working call center is the best test platform for an agent application.

If you are developing an agent application without access to a working Proactive Contact, Proactive Contact test systems are available that simulate call center activity.

---

You can also use the sample code provided on the Agent API CD-ROM to test your application. The sample code is for testing only and should not be used as a basis for production development.

For more information, contact your Proactive Contact representative.

## **Agent Binary**

The agent binary options, **-d** and **-t**, determine how the agent process starts. The default option is **-d**, which instructs Proactive Contact to start the agent process when Proactive Contact starts. The agent application connects to Proactive Contact through a Secure Socket Layer (SSL). The process runs as a daemon, starting an instance of the agent application for each agent at log in.

We do not recommend using the **-t** option without first conferring with your Proactive Contact representative. When appropriate, use the **-t** option to start the agent binary manually during development or testing. To manually start the agent binary on Proactive Contact, open a command line interface to Proactive Contact, type the command `agent -t`, and press Enter.

## **Agent API Messages**

The agent application and the agent binary exchange a series of transactions that control the agent's work session. The `Moagent32.log` provides a record of these transactions. Each message type has specific parameters.

### **Command Messages (C)**

Command messages take from zero to four parameters. The data segment consists of the parameters.

### **Response Messages (R)**

Response messages always receive the generic complete message (Proactive Contact code M00000) when the command finishes processing. Many commands receive the pending message (S28833). It indicates that the command is acknowledged but requires further processing by Proactive Contact.

### **Data Messages (D)**

Data messages consist of an Proactive Contact message and the requested data. The Proactive Contact message will be M00000 indicating the request was completed or M00001 indicating that more data is to be returned in subsequent messages.

The data segment number indicates the number of data pieces being returned.

### **Notification Event Messages (N)**

Notification event messages consist of an Proactive Contact message and the notify data.

## Agent Application Initiated Messages

The agent application initiates most transactions with a Command message. The agent binary replies with one or more of the following messages:

P	Pending message that indicates the command has been received but requires additional processing by Proactive Contact.
D	Data message that returns the data requested by the agent application.
R	Response message that indicates success or failure of the command. All commands from the agent application receive a response message.
B	Busy message that indicates that Proactive Contact cannot process the command. These messages are extremely rare.

## Agent Binary Initiated Messages

The only transactions that the agent binary initiates are notification event messages. The agent application responds to appropriate handling in order to execute another function. It does not respond to notification event messages.

## Message Formats

Messages consist of a header and data. While the header format for a particular message is always the same, the data format depends on where the message originated, the agent application or the agent binary.

### Message Headers

Message headers are fixed at 55 bytes long. All agent binary commands and notification events begin with AGT. The following table lists the components of each header field, including the size and format of the field.

---

Each field is left justified and right-padded with space characters.

Field	Description	Length
Keyword	A unique identifier for agent commands and events.	20 bytes

Field	Description	Length
Type	Command or event message type. <b>C</b> - Command from agent application to agent binary <b>P</b> - Pending response from agent binary to agent application <b>D</b> - Data message from agent binary to agent application <b>R</b> - Response from agent binary to agent application <b>B</b> - Busy message from agent binary to agent application (very rare) <b>N</b> - Notification event from agent binary to agent application	1 byte
Client	Reserved for future use. "Client" appears in the message header.	20 bytes
Process ID	Reserved for future use. "ProcID" appears in the message header.	6 bytes
Invoke ID	Reserved for future use. "InvokeID" appears in the message header.	4 bytes
Number of segments	Number of data segments in the data portion of the record.	4 bytes

For example, the header for AGTLogon, a connection command message, might look like the following.

Keyword	Type	Client	Process ID	Invoke ID	Number of segments
AGTLogon	C	Client	ProcID	Invoke ID	3
20 bytes	1 byte	20 bytes	6 bytes	4 bytes	4 bytes

AGTAdjustHeadset is an Agent application command message. It does not require the process to return data. The return message is Response with no data and M00000.

AGTListJobs is a system command that returns a list of active jobs.

Following are two examples of AGTListJobs messages. The first example is the agent application to the agent binary. The second example is the agent binary to the agent application.

## Agent Application to Agent Binary

```
0001 AGTListJobs
0002 C
0003 tester
0004 11111
0005 18
0006 1 (Message data separator, ASCII x 1E)
0007 A (Message terminator (ETX), ASCII x03)
```

The data parameter for the command is JobType.

The following table lists acceptable parameters.

Parameter	Definition
A	All
B	Blend
O	Outbound
I	Inbound
M	Managed dialing

## Agent Binary to Agent Application

```
AGTListJobs          D Agent server15395 18 8$0$ M00001$
                     B,blend1,I $ I,inbnd1,I $ M,managed,I $
                     O,outbnd,I $ O,outbnd2,A $
                     O,outbndtest,I+
```

The 8 in the data portion indicates that there are eight segments in the data field. The data segments are:

```
0 $
M00001 $
B,blend1,I $
I,inbnd1,I $
M,managed,I $
O,outbnd,I $
O,outbnd2,A$
O,outbndtest,I+
```

The return value includes M00001 indicating data being returned. The parameters for the returned data are:

```
<JobType>, <JobName>, <Status> <JobType>, <JobName>, <Status> ...#
```

The command status indicator is a single-digit numeric indicator of the command status on Proactive Contact. Successful commands return a 0 status. Errors return a 1 status.

Message numbers are Proactive Contact internal identifiers used by the Agent API to find message texts in the Moagent32.ini file.

Proactive Contact message codes are one alpha character followed by five numbers.

The alpha character at the beginning of the Proactive Contact message code indicates the message type.

Parameter	Definition
M	Message
E	Error
S	Status

There are two acceptable status types.

Parameter	Definition
I	Inactive
A	Active

(In the preceding data message, the parameter line “O,outbnd2,A” represents the only active job. The job name is outbnd2.)

The following table shows the process message format for the data portion of the message and matches it with the example.

Separator	Status (0 or 1)	Separator	Message number	Separator	Data	Message terminator
1 byte	1 byte	1 byte	6 bytes	1 byte	Variable length	1 byte



## Message Data Separators

Message data separators delimit record fields. The data separator for a socket connection is an ASCII x1E (RS).

## Message Terminators

Message terminators end each record. There are two kinds of message terminators: Incomplete and Complete.

An incomplete message continues in the next message. Continuation messages only occur in messages sent from the agent binary to the agent application. The message terminator is an ASCII x17 (ETB).

If the message is complete, the message terminator for a socket connection is an ASCII x03 (ETX).

## Data from Agent Application to Agent Binary

Agent application data consists of an initial message separator, data segments (field records), and the message terminator.

The following table shows the size of each field in the data portion of the message and matches each field to the AGTUpdateField example.

	Separator	Data	Message terminator
Length	1 byte [x1E]	Variable length segment [x1E] segment [x1E] segment...	1 byte [x03] or [x17]
Example	§	0 § AREA2 § 206	‡

## Data from Agent Binary to Agent Application

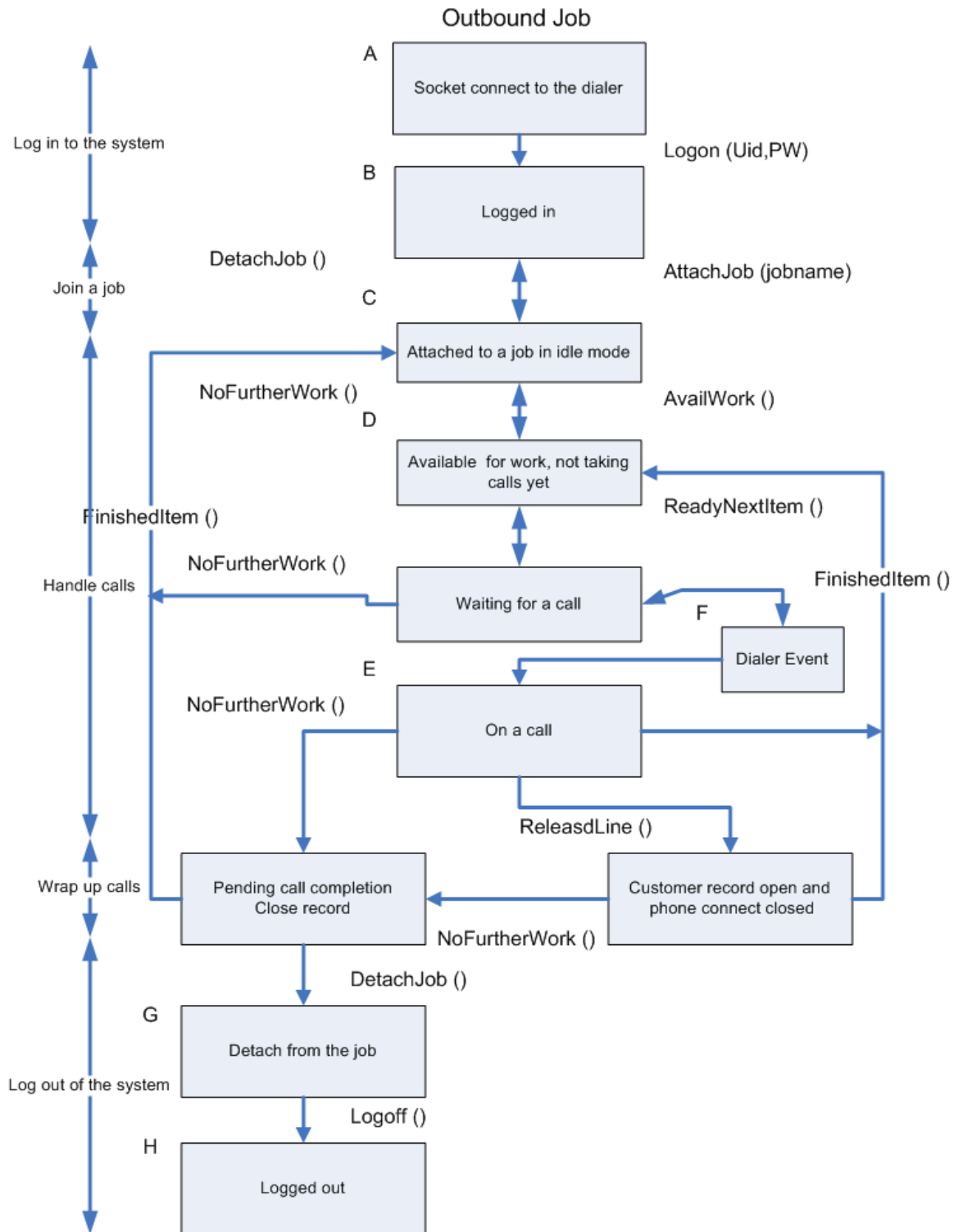
For more information about message numbers and the DLL, see [Using Proactive Contact Agent API DLL](#).

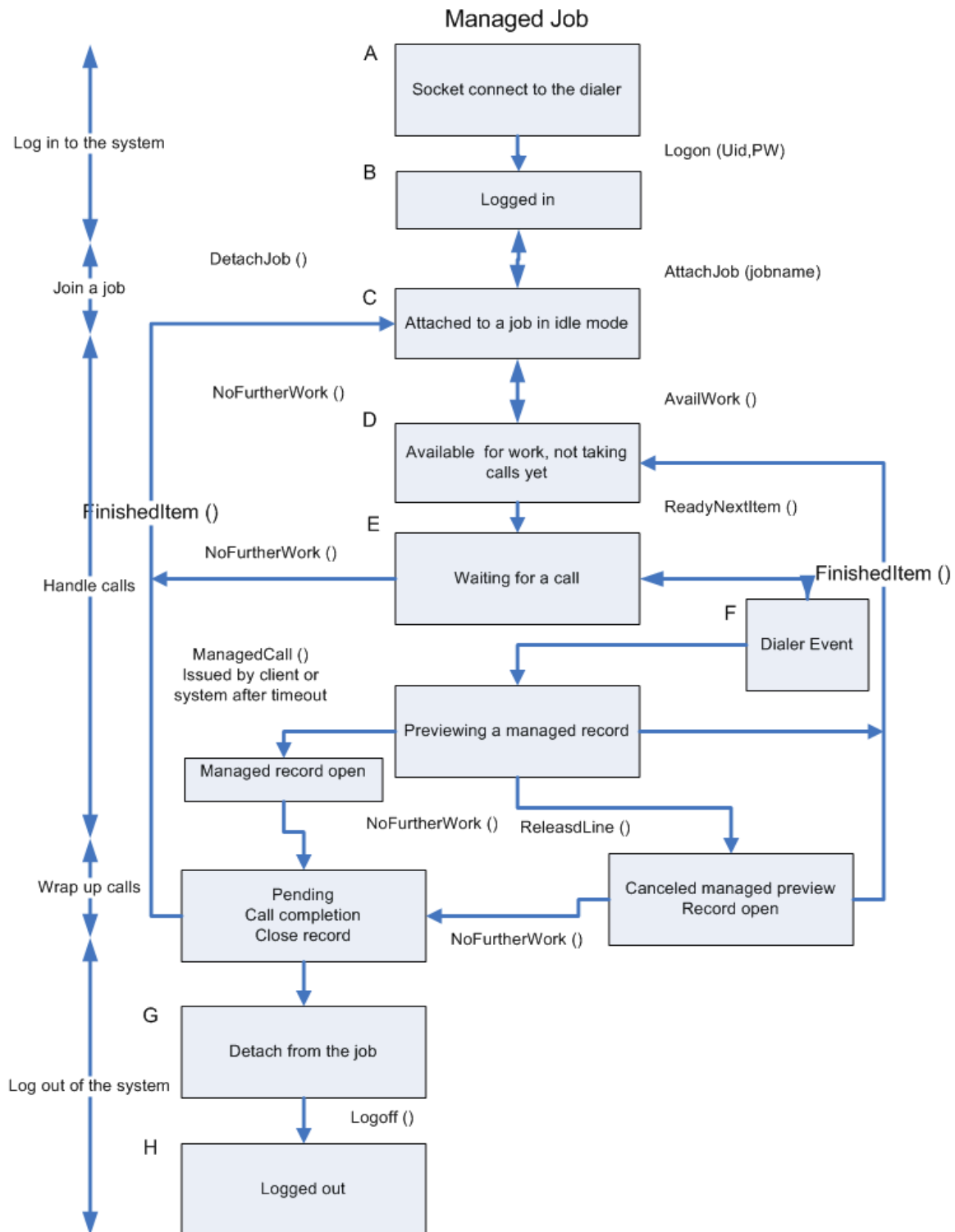
## Agent States

After the agent application establishes a connection with the agent binary, the agent workstation is ready for the agent to begin work. Agents progress through a series of states on Proactive Contact. Agent states can include:

- Not logged in to Proactive Contact
- Logged in to Proactive Contact but not attached to a job
- Attached to a job but not available for work
- Attached to a job, available for work, but not ready for a call
- Waiting for a call
- On a call

Agent API provides commands that move the agent from one state to another. The following diagrams show the various agent states during outbound calling. The list following the Managed Dialing Job diagram details the commands you can issue using the corresponding interface method in each state, labeled A - H.





When an agent selects a job, the agent application attaches the job. When the agent joins a job, the agent application logs the agent on to the job.

## Outbound Job Commands (by agent state)

The following tables describe the outbound job commands by agent state:

Commands Available	Agent State(s)
State "A" commands	Logon
State "B" commands	SendMessage LogonAcq (Agent Blending systems only, using Predictive Agent Blending or Proactive Agent Blending) LogloStart, LogloStop ReserveHeadset, ConnHeadset EchoOn, EchoOff (-t option only) AttachJob DumpData, SetWorkClass ListCallFields, ListCallLists, ListState, ListJobs Logoff
State "C" commands	AvailWork ClearDataSet DisconnHeadset, FreeHeadset ListCallbackFmt, ListDataFields, ListKeys, ListUnits SetDataField, SetNotifyKeyFields, SetUnit, SetWorkClass DetachJob
State "D" commands	ReadyNextItem
State "E" commands	ManagedCall, PreviewRecord (Managed Dialing job only) UpdateField, ReadField MoFlashSupv, MoFlashBlind, DialDigit, TransferCall HangupCall, HoldCall, ManualCall, HookFlashLine DoNotCall FinishedItem, NoFurtherWork, ReleaseLine SetCallBack AdjustHeadset, GetHeadSetVol
State "F" events from Proactive Contact	ReceiveMessage, SystemError, CallNotify HeadsetConnBroken, JobEnd, UnitEnd, TiicbAbort, TiicbFeNotify, TiicbOffline, TiicbOnline JobTransLink, JobTransRequest
State "G" commands	DetachJob

Commands Available	Agent State(s)
State "H" commands	Logoff LogoffAcid (Agent Blending systems only, using Predictive Agent Blending or Proactive Agent Blending)

## Outbound Job Message Scenarios

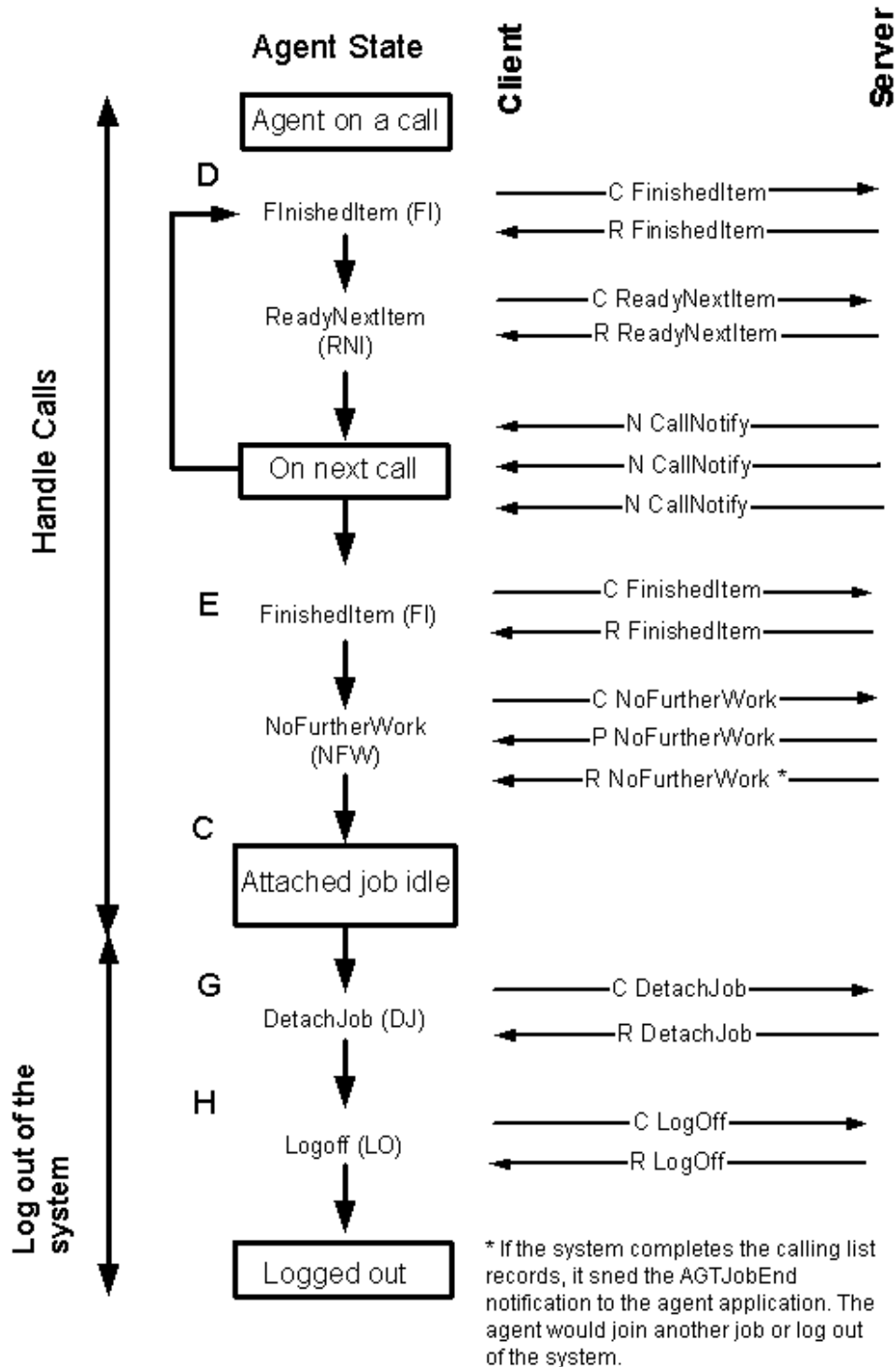
The following diagrams show sample message scenarios during outbound calling when an agent requests to log out of Proactive Contact. Prefixes precede commands and notification events: "C" identifies a command, "R" a response indicating the success or failure of a command, "P" a pending command requiring more processing by the server, and "N" a notification event.

The first diagram shows messages when an agent requests to log out an outbound job where agents remain on the job.

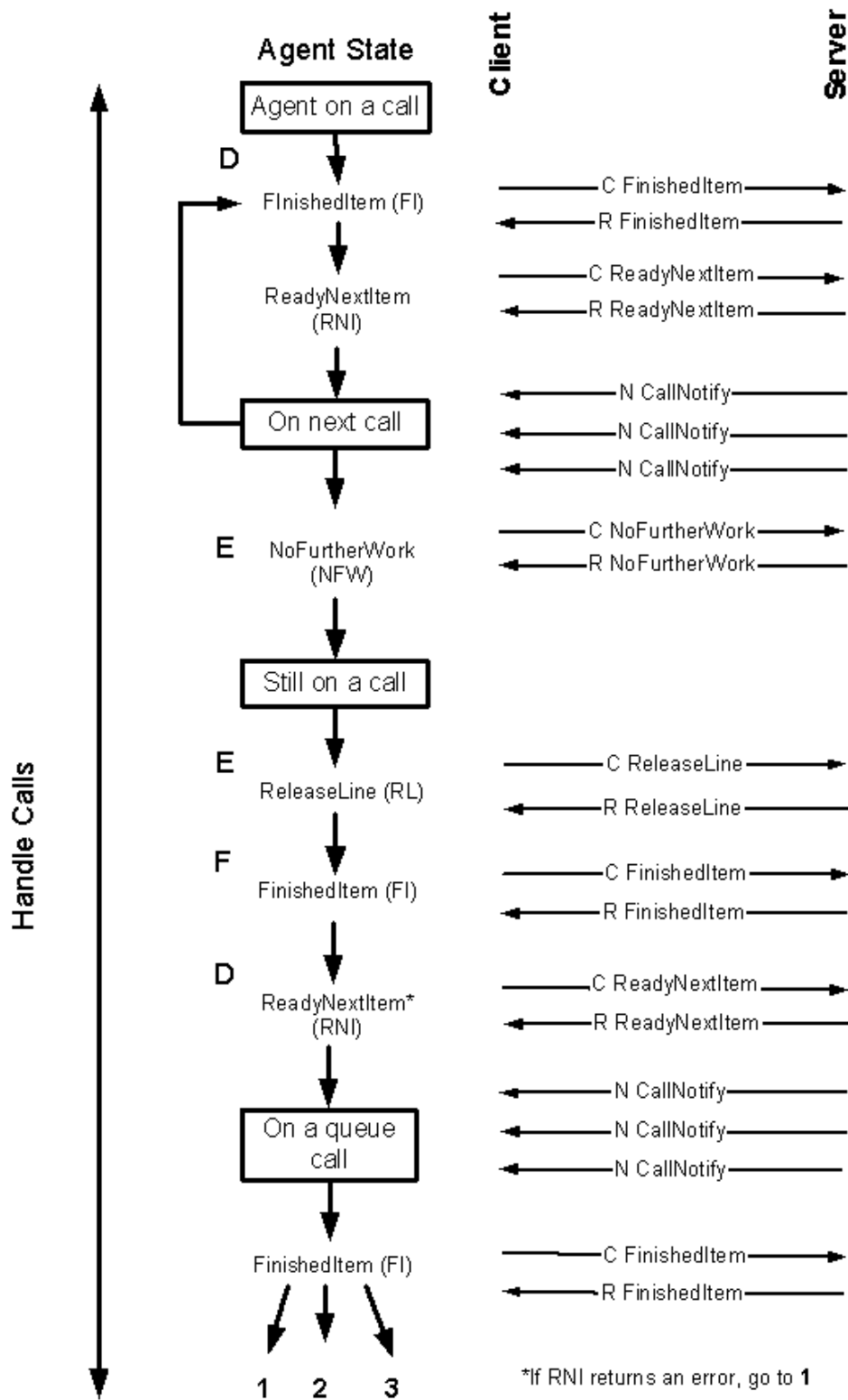
The second diagram shows messages when the agent requesting to log out of the outbound job is the last agent remaining on the job.

The third diagram shows messages when an agent requests to log out of a blend job on a Proactive Contact Intelligent Call Blending system.

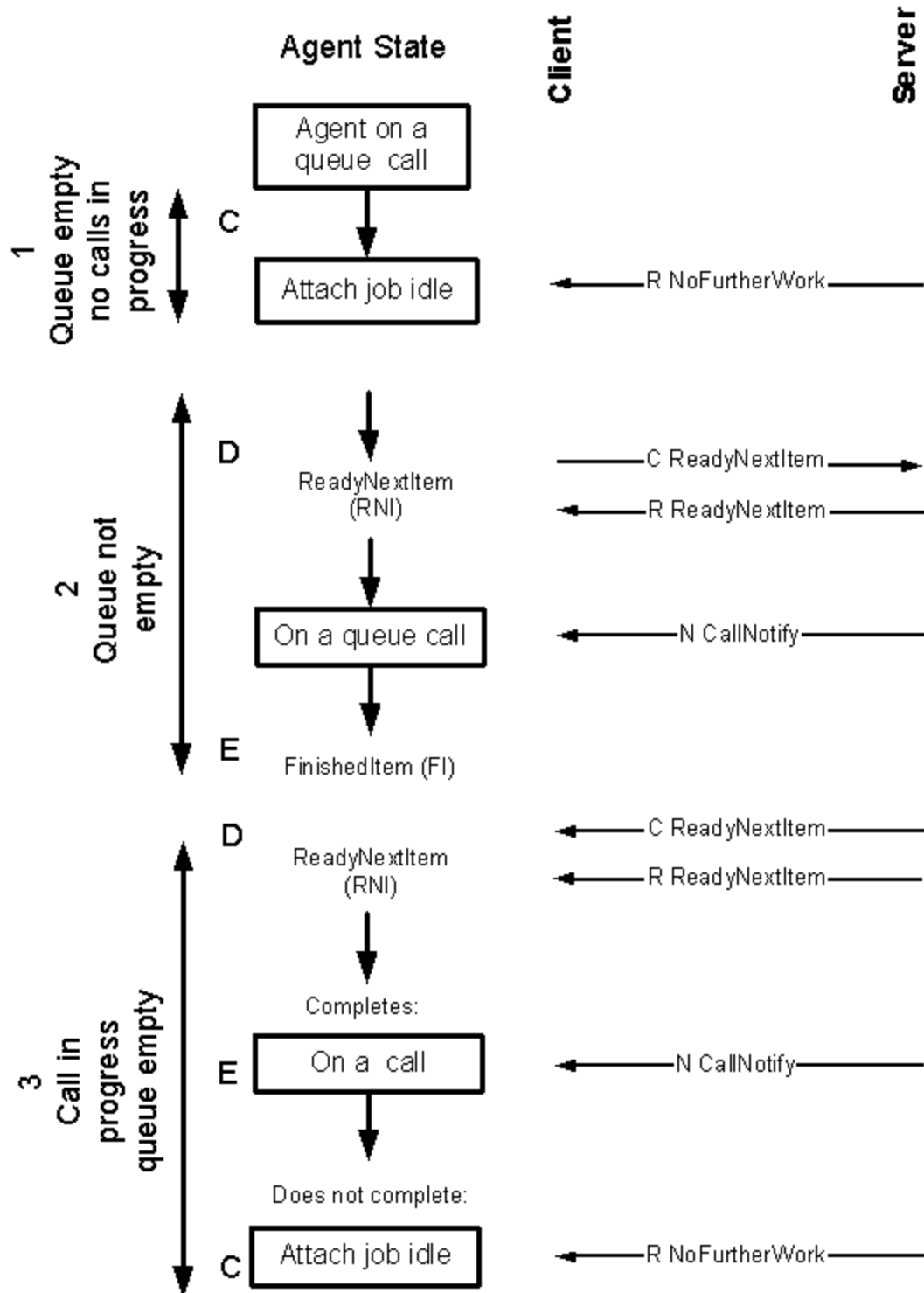
## Additional Agents Remain on the Outbound Job



## Last Outbound Agent on the Job Requests to Log Out

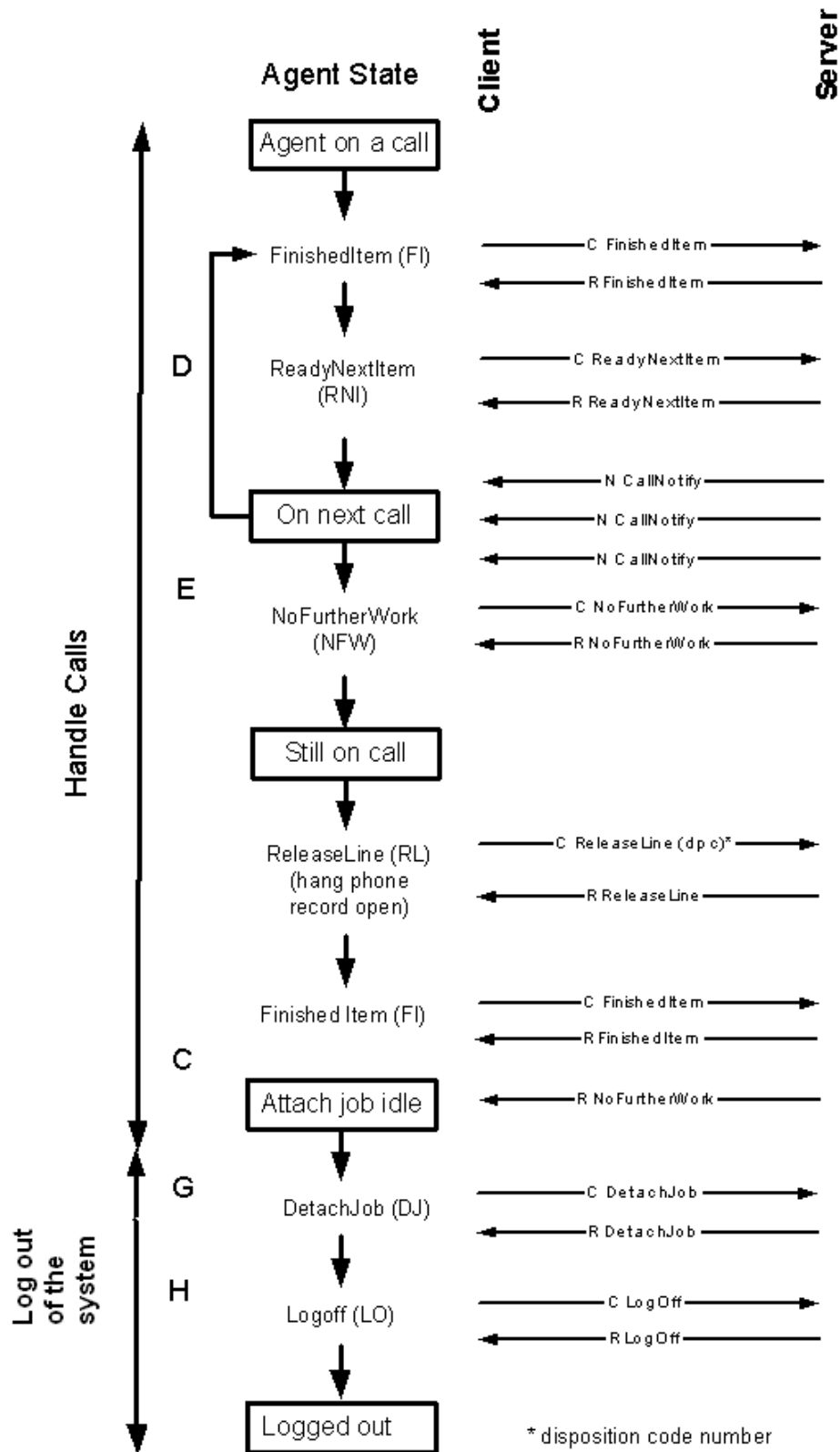


## Last Outbound Agent on the Job Requests to Log Out (continued)





## Agent Requests to Log Out of a Blend Job



## Moagent32.log

Use Moagent32.log to track the flow of your application. The log file receives records from Moagent32.dll until you close the agent application. The agent application labels each message and event notification with a time stamp and an indicator of the message's direction: from the agent binary to the client (Client <) and from the client to the agent binary (Client >). When the message originates from the agent application a connection command might look like the following:

```
Client > 11:35:55 AGTLogon C102 203 304 3
04
****
```

### PCAPI\_5.1.0.0.4

When the message originates from the agent binary, a connection command may look like either of the following:

```
Client < 11:35:55 AGTLogon PAgent server 1539 304 2
0
S28833
Client < 11:35:55 AGTLogon RAgent server 1539 304 2
0
M00000
```

You can set properties for the Moagent32.log file from your agent application. See [Using Proactive Contact Agent API DLL](#) for information on setting default properties (including code samples). Log file properties you can set include whether to create a log file (default is Yes), file name, and file location.

## Sample Agent Application Session

The agent logs in to Proactive Contact, selects a job, performs job setup activities, joins a job, and handles calls. When finished working, the agent logs off Proactive Contact. The agent application terminates the session

The following Moagent32.log file example begins with the agent application establishing a socket connection to the agent binary.

This example is explained line-for-line beginning half-way through the example.

```
Client > 11:35:53 Connection Parameters
<quillcene - your Avaya Proactive Contact server name>
agent
04
****
4
1 Client < 11:35:55 AGTSTART NAgent server 1539 0 2
0
AGENT_STARTUP
2 Client > 11:35:55 AGTLogon C102 203 304 3
04
****
PCAPI_5.1.0.0.4
3 Client < 11:35:55 AGTLogon PAgent server 1539 304 2
0
S28833
4 Client < 11:35:55 AGTLogon RAgent server 1539 304 2
0
M00000
5 Client > 11:35:55 AGTReserveHeadset C102 203 304 1
```

## Avaya Proactive Contact Agent API 5.2

```
4
6 Client < 11:35:55 AGTReserveHeadset PAgent server 1539 304 2
0
S28833
7 Client < 11:35:56 AGTReserveHeadset RAgent server 1539 304 2
0
M00000
8 Client > 11:35:56 AGTConnHeadset C102 203 304 0
9 Client < 11:35:56 AGTConnHeadset PAgent server 1539 304 2
S28833
10 Client < 11:35:57 AGTConnHeadset RAgent server 1539 304 2
0
M00000
11 Client > 11:36:37 AGTListJobs C100 200 300 1
A
12 Client < 11:36:37 AGTListJobs DAgent server 1539 300 14
0
M00001
B,blend,I
I,inbnd,I
I,inbnd1,I
O,infinity,I
M,managed,A
O,out2unit,A
O,outbnd,A
O,outbnd1,A
O,outbnd2,A
O,sl,I
O,verify,I
O,virtual,I
13 Client < 11:36:37 AGTListJobs RAgent server 1539 300 2
0
M00000
14 Client > 11:37:12 AGTSetWorkClass C100 200 300 1
0
15 Client < 11:37:12 AGTSetWorkClass RAgent server 1539 300 2
0
M00000
16 Client > 11:39:02 AGTAttachJob C100 200 300 1
outbnd1
17 Client < 11:39:02 AGTAttachJob RAgent server 1539 300 2
0
M00000
18 Client > 11:39:50 AGTSetNotifyKeyFieldC100 200 300 2
0
ACCTNUM
19 Client < 11:39:50 AGTSetNotifyKeyFieldRAgent server 1539 300 2
0
M00000
20 Client > 11:40:19 AGTSetDataField C100 200 300 2
0
BAL
21 Client < 11:40:19 AGTSetDataField RAgent server 1539 300 2
0
M00000
22 Client > 11:40:30 AGTSetDataField C100 200 300 2
0
CREDLINE
```

## Avaya Proactive Contact Agent API 5.2

```
23 Client < 11:40:30 AGTSetDataField RAgent server 1539 300 2
0
M00000
```

Line	Meaning
1	The agent application connection to Proactive Contact is complete. The agent binary sends the start-up message, indicating it is ready for an agent to log in to the system.
2-4	When commands require additional processing by Proactive Contact, the agent binary issues a code S28833 message. This indicates the command is pending.
5-10	The agent application reserves and connects to a numbered headset on Proactive Contact. This number represents a key code or an extension, depending on Proactive Contact and the type of headset connection.
11-13	The agent application gets a list of all jobs on Proactive Contact. Each message shows the job type, name, and status (A for Active, I for Inactive). When a response includes data, the agent binary returns a data message followed by a response message.
14-15	Agent is setting the workclass to outbound.
16-17	The agent selects the outbound job. The agent application informs Proactive Contact.
18-23	To select fields, the agent application must know the calling list record structure.

```
24 Client > 11:41:27 AGTAvailWork C100 200 300 0
25 Client < 11:41:27 AGTAvailWork PAgent server 1539 300 2
0
S28833
26 Client < 11:41:27 AGTAvailWork RAgent server 1539 300 2
0
M00000
27 Client > 11:41:35 AGTReadyNextItem C100 200 300 0
28 Client < 11:41:36 AGTReadyNextItem RAgent server 1539 300 2
0
M00000
29 Client < 11:41:56 AGTCallNotify NAgent server 1539 0 5
0
M00001
JOHN DOE
OUTBOUND
ACCTNUM,5300292201447702
30 Client < 11:41:56 AGTCallNotify NAgent server 1539 0 4
0
M00001
BAL,0.00
CREDLINE,00
31 Client < 11:41:56 AGTCallNotify NAgent server 1539 0 2
0
M00000
32 Client > 11:43:00 AGTReadField C100 200 300 2
```

## Avaya Proactive Contact Agent API 5.2

```
O
PHONE2
33 Client < 11:43:01 AGTReadField DAgent server 1539 300 3
O
M00001
PHONE2,N,10,0000000000
34 Client < 11:43:01 AGTReadField RAgent server 1539 300 2
O
M00000
35 Client > 11:43:18 AGTReadField C100 200 300 2
O
AREA2
36 Client < 11:43:18 AGTReadField DAgent server 1539 300 3
O
M00001
AREA2,N,3,000
37 Client < 11:43:18 AGTReadField RAgent server 1539 300 2
O
M00000
```

Line	Meaning
------	---------

- |       |   |
|-------|---|
| 24-26 | The agent joins the job. The agent application logs in to the job with the AGTAvailWork command.  |
| 27-28 | The agent application indicates to Proactive Contact that the agent is ready to receive a call.   |
| 29-31 | Proactive Contact sends a customer record and an outbound call to the client. The agent application receives the requested fields specified in lines 18 through 23: the customer name, account number, account balance, and credit line. The agent application displays this information on the agent's screen while Proactive Contact connects the agent to the client's call. |
| 32-37 | The agent requests additional information from the customer record (the client's second phone number and area code). Proactive Contact returns the information, and the agent application displays it.  |

```
38 Client > 11:44:16 AGTUpdateField C100 200 300 3
O
AREA2
425
39 Client < 11:44:16 AGTUpdateField RAgent server 1539 300 2
O
M00000
40 Client > 11:44:37 AGTUpdateField C100 200 300 3
O
PHONE2
5551212
41 Client < 11:44:37 AGTUpdateField RAgent server 1539 300 2
O
M00000
42 Client > 11:44:53 AGTListCallbackFmt C100 200 300 0
43 Client < 11:44:53 AGTListCallbackFmt DAgent server 1539 300 4
O
M00001
```

## Avaya Proactive Contact Agent API 5.2

```
CCYY/MM/DD
2
44 Client < 11:44:53 AGTListCallbackFmt RAgent server 1539 300 2
0
M00000
45 Client > 11:45:42 AGTSetCallback C102 203 304 4
2002/02/26
05:00p
2
W. Gates
46 Client < 11:45:42 AGTSetCallback RAgent server 1539 304 2
1
E28842,B
47 Client > 11:45:55 AGTListCallbackFmt C100 200 300 0
48 Client < 11:45:55 AGTListCallbackFmt DAgent server 1539 300 4
0
M00001
CCYY/MM/DD
2
49 Client < 11:45:55 AGTListCallbackFmt RAgent server 1539 300 2
0
M00000
50 Client > 11:46:20 AGTSetCallback C102 203 304 4
2002/02/26
05:00p
1
W. Gates
```

Line	Meaning
------	---------

38-41	The agent updates the second phone number and area code. The agent application sends the new information to Proactive Contact. (To request and update information from the customer record, the agent application must know the calling list fields.)
-------	---

42-51	The agent tells Proactive Contact to call the customer again on February 26 at 5:00PM on the client's second phone. The format of the date must match the format previously supplied by AGTListCallbackFmt.
-------	---

```
51 Client < 11:46:20 AGTSetCallback RAgent server 1539 304 2
0
M00000
52 Client > 11:46:44 AGTReleaseLine C100 200 300 0
53 Client < 11:46:44 AGTReleaseLine PAgent server 1539 300 2
0
S28833
54 Client < 11:46:45 AGTReleaseLine RAgent server 1539 300 2
0
M00000
55 Client > 11:47:14 AGTFinishedItem C100 200 300 1
19
56 Client < 11:47:14 AGTFinishedItem RAgent server 1539 300 2
0
M00000
57 Client > 11:47:21 AGTReadyNextItem C100 200 300 0
58 Client < 11:47:21 AGTReadyNextItem RAgent server 1539 300 2
```

## Avaya Proactive Contact Agent API 5.2

```
0
M00000
59 Client < 11:47:41 AGTCallNotify NAgent server 1539 0 5
0
M00001
JANE SMITH
OUTBOUND
ACCTNUM,5300292201439436
60 Client < 11:47:41 AGTCallNotify NAgent server 1539 0 4
0
M00001
BAL,0.00
CREDLINE,00
61 Client < 11:47:41 AGTCallNotify NAgent server 1539 0 2
0
M00000
```

Line	Meaning
------	---------

- |       |  |
|-------|--|
| 52-54 | The agent releases the telephone line. This tells Proactive Contact to hang up the call and use the line for another call. Notice that releasing the telephone line does not release the customer record. (To select a script label (such as call_complete), the agent application must know the contents of the script file.) |
| 55-56 | The agent finishes working on the customer record and sets a completion code of 19 (recall). Proactive Contact contains a completion codes configuration for each job. (To select a completion code, the agent application must know the codes used by the job.)   |
| 57-58 | The agent application lets Proactive Contact know the agent is ready for another call.   |
| 59-61 | Proactive Contact sends a new customer record and an outbound call.  |

```
62 Client > 12:01:51 AGTDetachJob C100 200 300 0
63 Client < 12:01:51 AGTDetachJob RAgent server 1539 300 2
0
M00000
64 Client > 12:02:03 AGTDisconnHeadset C100 200 300 0
65 Client < 12:02:03 AGTDisconnHeadset PAgent server 1539 300 2
0
S28833
66 Client < 12:02:03 AGTDisconnHeadset RAgent server 1539 300 2
0
M00000
67 Client > 12:02:08 AGTFreeHeadset C100 200 300 0
68 Client < 12:02:08 AGTFreeHeadset RAgent server 1539 300 2
0
M00000
69 Client > 12:02:12 AGTLogoff C100 200 300 0
70 Client < 12:02:12 AGTLogoff RAgent server 1539 300 2
0
M00000
```

<b>Line</b>	<b>Meaning</b>
62-68	The agent indicates he or she is logging out. The agent application disconnects the headset and frees the headset ID. The agent is still logged on to Proactive Contact but cannot take customer calls.
69-70	The agent logs off Proactive Contact.

## Sample Managed Dialing Job

This example shows a portion of a session where an agent works on a Managed Dialing job. The agent becomes a Managed agent, selects a Managed Dialing job, and works on the job. The example begins with the agent logging in to Proactive Contact.

```
Client > 12:08:47 Connection Parameters
<quilcene - your Avaya Proactive Contact server name>
agent
04
****
4
1 Client < 12:08:49 AGTSTART NAgent server 1578 0 2
0
AGENT_STARTUP
2 Client > 12:08:49 AGTLogon C102 203 304 3
04
****
PCAPI_5.1.0.0.4
3 Client < 12:08:49 AGTLogon PAgent server 1578 304 2
0
S28833
4 Client < 12:08:49 AGTLogon RAgent server 1578 304 2
0
M00000
5 Client > 12:08:49 AGTReserveHeadset C102 203 304 1
4
6 Client < 12:08:49 AGTReserveHeadset PAgent server 1578 304 2
0
S28833
7 Client < 12:08:49 AGTReserveHeadset RAgent server 1578 304 2
0
M00000
8 Client > 12:08:49 AGTConnHeadset C102 203 304 0
9 Client < 12:08:49 AGTConnHeadset PAgent server 1578 304 2
0
S28833
10 Client < 12:08:51 AGTConnHeadset RAgent server 1578 304 2
0
M00000
11 Client > 12:10:49 AGTSetWorkClass C100 200 300 1
M
12 Client < 12:10:49 AGTSetWorkClass RAgent server 1578 300 2
0
M00000
13 Client > 12:11:42 AGTListJobs C100 200 300 1
```



## Avaya Proactive Contact Agent API 5.2

```
M
14 Client < 12:11:42 AGTListJobs DAgent server 1578 300 3
0
M00001
15 Client < 12:11:42 AGTListJobs RAgent server 1578 300 2
0
M00000
16 Client > 12:11:47 AGTAttachJob C102 203 304 1
managed
17 Client < 12:11:47 AGTAttachJob RAgent server 1578 304 2
0
M00000
```

Line	Meaning
1-4	The agent application connection to Proactive Contact is complete. The agent binary sends the start-up message, indicating it is ready for an agent to log in to the system. The agent logs on to the system.
5-10	The agent application reserves and connects to a numbered headset on Proactive Contact. This number represents a key code or an extension, depending on Proactive Contact and the type of headset connection.
11-12	The agent selects the managed agent type. The agent application sends the agent type to Proactive Contact.
13-17	The agent selects a Managed Dialing job. The agent application attaches the job.

```
18 Client > 12:12:14 AGTListDataFields C100 200 300 1
0
19 Client < 12:12:14 AGTListDataFields DAgent server 1578 300 50
0
M00001
SYSNUM,4,N,F
PRIN,4,C,F
CCODE,3,C,F
ACCTNUM,16,N,F
NAME,26,C,F
NAME2,26,C,F
CBFLAG,1,C,F
PHONE2,10,N,F
AREA2,3,N,F
PHONE1,10,N,F
AREA,3,N,F
EXTERNAL,1,C,F
INTERNAL,1,C,F
BAL,10,$,F
CREDLINE,7,$,F
DELQUENT,10,$,F
DAYS,3,N,F
PAYDAY,10,D,F
PAYAMT,8,$,F
ZIPCODE,5,N,F
BEHSCORE,3,C,F
AGENT,8,C,F
DTE,10,D,F
TME,8,T,F
```

## Avaya Proactive Contact Agent API 5.2

CODE,2,C,F  
ENTRYDATE,10,D,F  
STATUSFLAG,1,C,F  
RECALLDATE,10,D,F  
RECALLTIME,8,T,F  
DAYSCNT,3,N,F  
PHONESTAT,2,C,F  
ZONEPHONE1,1,C,F  
ZONEPHONE2,1,C,F  
PHONECNT1,2,N,F  
PHONECNT2,2,N,F  
CURPHONE,2,N,F  
RECALLPHONE,2,C,F

Line	Meaning
------	---------

18-20	The agent application gets field definitions from the calling list for fields that are part of each customer record on this job. The command specifies a calling list type of OUTBOUND.
-------	---

COUNTER,3,N,F  
DUR4,9,N,F  
DUPE,1,C,F  
JOBNAME,20,C,F  
FINOPER,8,C,F  
SVJCODE,2,C,F  
JOBID,8,C,F  
SHADOWJOB,20,C,F  
RECALLNUMBER,12,C,F  
MASTERZONE,1,C,F  
DUPEREC,10,C,F  
20 Client < 12:12:15 AGTListDataFields RAgent server 1578 300 2  
0  
M00000  
21 Client > 12:12:35 AGTSetNotifyKeyFieldC100 200 300 2  
0  
ACCTNUM  
22 Client < 12:12:35 AGTSetNotifyKeyFieldRAgent server 1578 300 2  
0  
M00000  
23 Client > 12:12:48 AGTSetDataField C100 200 300 2  
0  
CREDLINE  
24 Client < 12:12:48 AGTSetDataField RAgent server 1578 300 2  
0  
M00000  
25 Client > 12:13:14 AGTAvailWork C100 200 300 0  
26 Client < 12:13:15 AGTAvailWork PAgent server 1578 300 2  
0  
S28833  
27 Client < 12:13:15 AGTAvailWork RAgent server 1578 300 2  
0  
M00000  
28 Client > 12:13:21 AGTReadyNextItem C100 200 300 0  
29 Client < 12:13:21 AGTReadyNextItem RAgent server 1578 300 2  
0  
M00000

## Avaya Proactive Contact Agent API 5.2

Line	Meaning
21-22	The agent application sets the key field to include record preview event notification.
23-24	The agent application sets a data field to send with the second preview event notification message.
25-29	The agent logs in to the job. The agent application tells Proactive Contact the agent is ready to receive the first preview record.

```
30 Client < 12:13:23 AGTPreviewRecord NAgent server 1578 0 5
0
M00001
JOHN DOE (Preview)
MANAGED
ACCTNUM,5300292201411260
31 Client < 12:13:23 AGTPreviewRecord NAgent server 1578 0 3
0
M00001
CREDLINE,00
32 Client < 12:13:23 AGTPreviewRecord NAgent server 1578 0 2
0
M00000
33 Client < 12:14:23 AGTManagedCall C100 200 300 0
0
34 Client < 12:14:23 AGTManagedCall PAgent server 1578 300 2
0
S28833
35 Client < 12:14:35 AGTManagedCall DAgent server 1578 300 3
0
M00001
(CONNECT)
36 Client < 12:14:35 AGTManagedCall RAgent server 1578 300 2
0
M00000
37 Client > 12:20:25 AGTUpdateField C100 200 300 3
0
PAYDAY
12/15/2001
38 Client < 12:20:26 AGTUpdateField RAgent server 1578 300 2
0
M00000
39 Client > 12:20:51 AGTUpdateField C100 200 300 3
0
PAYAMT
500
40 Client < 12:20:51 AGTUpdateField RAgent server 1578 300 2
0
M00000
```

Line	Meaning
30-32	The agent application sends an AGTPreviewRecord command to tell Proactive Contact that the agent can preview the record.

33-36 The Proactive Contact preview time elapses. The system makes the call.

37-40 During the conversation with the client, the agent updates the client's next payment date and the amount the customer will pay on that date. The agent application sends the new information to Proactive Contact. To update the customer record, the agent application must know the fields included with each customer record in the job.

```

41 Client > 12:22:09 AGTFinishedItem C100 200 300 1
23
42 Client < 12:22:09 AGTFinishedItem RAgent server 1578 300 2
0
M00000
43 Client > 12:22:17 AGTReadyNextItem C100 200 300 0
44 Client < 12:22:17 AGTReadyNextItem RAgent server 1578 300 2
0
M00000
45 Client < 12:22:18 AGTPreviewRecord NAgent server 1578 0 5
0
M00001
MICHAEL SMITH (Preview)
MANAGED
ACCTNUM,5300292201410379
46 Client < 12:22:19 AGTPreviewRecord NAgent server 1578 0 3
0
M00001
CREDLINE,00
47 Client < 12:22:19 AGTPreviewRecord NAgent server 1578 0 2
0
M00000
48 Client > 12:22:51 AGTFinishedItem C100 200 300 1
22
49 Client < 12:22:51 AGTFinishedItem RAgent server 1578 300 2
0
M00000
50 Client > 12:23:03 AGTNoFurtherWork C100 200 300 0
51 Client < 12:23:03 AGTNoFurtherWork PAgent server 1578 300 2
0
S28833
52 Client < 12:23:03 AGTNoFurtherWork RAgent server 1578 300 2
0
M00000
53 Client > 12:23:09 AGTDetachJob C100 200 300 0
54 Client < 12:23:09 AGTDetachJob RAgent server 1578 300 2
0
M00000
55 Client > 12:23:14 AGTLogoff C100 200 300 0
56 Client < 12:23:14 AGTLogoff RAgent server 1578 300 2
0
M00000

```

Line	Meaning
------	---------

41-42	After the agent completes the call and the record, he or she selects a completion code. The agent application transmits the finished item and release line commands to Proactive Contact.
-------	---

- 43-44      The agent application tells Proactive Contact the agent is ready to receive the next preview record.
- 45-47      The agent application sends an AGTPreviewRecord command to tell Proactive Contact that the agent can preview the record.
- 48-49      The agent cancels the call.
- 50-56      The agent application completes the agent log out and ends the session, breaking the connection with Proactive Contact.

### Sample Unit Work List Job

This example shows the login procedure and job setup for a Unit Work List job. The agent joins the job as an outbound agent, and chooses a unit ID value. The example begins with the agent application establishing a connection to the agent binary on Proactive Contact.

```
Client > 11:05:18 Connection Parameters
<quillcene - your Avaya Proactive Contact server name>
agent
04
****
4
1 Client < 11:05:19 AGTSTART NAgent server 1577 0 2
0
AGENT_STARTUP
2 Client > 11:05:19 AGTLogon C102 203 304 3
04
****
PCAPI_5.1.0.0.4
3 Client < 11:05:19 AGTLogon PAgent server 1577 304 2
0
S28833
4 Client < 11:05:20 AGTLogon RAgent server 1577 304 2
0
M00000
5 Client > 11:05:20 AGTReserveHeadset C102 203 304 1
4
6 Client < 11:05:20 AGTReserveHeadset PAgent server 1577 304 2
0
S28833
7 Client < 11:05:21 AGTReserveHeadset RAgent server 1577 304 2
0
M00000
8 Client > 11:05:21 AGTConnHeadset C102 203 304 0
9 Client < 11:05:21 AGTConnHeadset PAgent server 1577 304 2
0
S28833
10 Client < 11:05:23 AGTConnHeadset RAgent server 1577 304 2
0
M00000
11 Client > 11:05:35 AGTListJobs C100 200 300 1
A
```

## Avaya Proactive Contact Agent API 5.2

```
12 Client < 11:05:35 AGTListJobs DAgent server 1577 300 15
0
M00001
B,blend,I
I,inbnd,I
I,inbnd1,I
O,infinity,I
M,managed,A
M,out1unit,I
O,out2unit,A
O,outbnd,I
O,outbnd1,A
O,outbnd2,A
O,sl,I
O,verify,I
O,virtual,I
13 Client < 11:05:36 AGTListJobs RAgent server 1577 300 2
0
M00000
14 Client > 11:05:39 AGTAttachJob C102 203 304 1
out2unit
15 Client < 11:05:39 AGTAttachJob RAgent server 1577 304 2
0
M00000
16 Client > 11:06:05 AGTSetNotifyKeyFieldC100 200 300 2
0
ACCTNUM
17 Client < 11:06:05 AGTSetNotifyKeyFieldRAgent server 1577 300 2
0
M00000
18 Client > 11:06:17 AGTSetDataField C100 200 300 2
0
BAL
19 Client < 11:06:17 AGTSetDataField RAgent server 1577 300 2
0
M00000
20 Client > 11:06:34 AGTSetDataField C100 200 300 2
0
CREDLINE
21 Client < 11:06:34 AGTSetDataField RAgent server 1577 300 2
0
M00000
```

Line	Meaning
1	The agent application connection to Proactive Contact is complete. The agent binary sends the start-up message, indicating it is ready for an agent to log in to the system.
2-4	The agent enters a user name and password. Proactive Contact accepts the agent log in. When commands require additional processing by Proactive Contact, the agent binary issues a message with a code of S28833. This indicates the command is pending.
5-10	The agent application reserves the headset ID and connects to headset number 4 on Proactive Contact. This number could represent a key code or an extension, depending on Proactive Contact and the type of headset connection.

- 11-13 The agent application gets a list of all jobs on Proactive Contact. Each message shows the job type, name, and status (A for Active, I for Inactive). The only active job is a Unit Work List job. The calling list type for the Unit Work List job is O (outbound). Unit Work List jobs are outbound jobs.
- 14-15 The agent selects the Unit Work List job. The agent application attaches the job.
- 16-17 The agent application selects the customer record key field for Proactive Contact to send when connecting a call to the agent.
- 18-21 The agent application selects two additional fields for Proactive Contact to send with call notifications.

```
22 Client > 11:06:43 AGTListUnits C100 200 300 0
23 Client < 11:06:43 AGTListUnits DAgent server 1577 300 3
0
M00001
9200
24 Client < 11:06:43 AGTListUnits RAgent server 1577 300 2
0
M00000
25 Client > 11:06:46 AGTSetUnit C102 203 304 1
9200
26 Client < 11:06:46 AGTSetUnit RAgent server 1577 304 2
0
M00000
27 Client > 11:07:00 AGTAvailWork C100 200 300 0
28 Client < 11:07:00 AGTAvailWork PAgent server 1577 300 2
0
S28833
29 Client < 11:07:00 AGTAvailWork RAgent server 1577 300 2
0
M00000
```

Line	Meaning
------	---------

- |       |  |
|-------|--|
| 22-24 | The agent application requests a listing of the unit ID values Proactive Contact is using on the current job. (The application displays the available unit ID values or the dialog box.) |
| 25-26 | The agent chooses unit ID value 9200. Proactive Contact will only connect the agent to clients who have a Unit Work List unit ID value of 9200.  |
| 27-29 | The agent joins the Unit Work List job to begin taking calls. The agent application tells Proactive Contact the agent is ready for a call.   |

## Sample Agent Owned Recall

This example does not show the login procedure and job setup for an Agent Owned Recall. The agent joins the job as an outbound agent.

## Avaya Proactive Contact Agent API 5.2

```
Client > 10:55:29 Connection Parameters
<quillcene - your Avaya Proactive Contact server name>
agent
04
****
4
1 Client > 10:56:24 AGTAvailWork C100 200 300 0
2 Client < 10:56:24 AGTAvailWork PAgent server 2035 300 2
0
S28833
3 Client < 10:56:24 AGTAvailWork RAgent server 2035 300 2
0
M00000
4 Client > 10:56:30 AGTReadyNextItem C100 200 300 0
5 Client < 10:56:30 AGTReadyNextItem RAgent server 2035 300 2
0
M00000
6 Client < 10:56:49 AGTCallNotify NAgent server 2035 0 5
0
M00001
JOHN DOE
OUTBOUND
ACCTNUM,5300292221318024
7 Client < 10:56:49 AGTCallNotify NAgent server 2035 0 4
0
M00001
PHONE1,2032699002
PHONE2,0000000000
8 Client < 10:56:49 AGTCallNotify NAgent server 2035 0 2
0
M00000
9 Client > 10:59:06 AGTListCallbackFmt C100 200 300 0
10 Client < 10:59:06 AGTListCallbackFmt DAgent server 2035 300 4
0
M00001
CCYY/MM/DD 2
11 Client < 10:59:06 AGTListCallbackFmt RAgent server 2035 300 2
0
M00000
12 Client > 10:59:23 AGTSetCallback C102 203 304 3
2002/03/05
00:30+
1
13 Client < 10:59:23 AGTSetCallback RAgent server 2035 304 2
0
M00000
14 Client > 10:59:39 AGTFinishedItem C100 200 300 1
98
15 Client < 10:59:39 AGTFinishedItem RAgent server 2035 300 2
0
M00000
16 Client > 10:59:55 AGTReadyNextItem C100 200 300 0
17 Client < 10:59:55 AGTReadyNextItem RAgent server 2035 300 2
0
M00000
-----
18 Client < 11:19:36 AGTCallNotify NAgent server 2035 0 5
0
```



## Avaya Proactive Contact Agent API 5.2

```
M00001
AgtOwnedRc: 04 J: outbnd1 U: Allid T: 11.26.00
OUTBOUND
ACCTNUM,5300292221375099
19 Client < 11:19:37 AGTCallNotify NAgent server 2035 0 3
0
M00001
PHONE1,2033879331
20 Client < 11:19:37 AGTCallNotify NAgent server 2035 0 2
0
M00000
21 Client > 11:20:36 AGTFinishedItem C100 200 300 1
22
22 Client < 11:20:36 AGTFinishedItem RAgent server 2035 300 2
0
M00000
23 Client > 11:20:41 AGTReadyNextItem C100 200 300 0
```

Line	Meaning
------	---------

- |       |   |
|-------|---|
| 1-5   | Agent is ready to begin taking calls on a job called outbnd1.   |
| 6-8   | Proactive Contact sends a customer record and an outbound call to the agent application.  |
| 9-13  | The agent sets a recall on the customer record.   |
| 14-15 | The agent releases the record with a completion code 98 indicating it is an Agent Owned Recall.   |
| 16-17 | The agent is ready to continue receiving calls on this job.   |
| 18-20 | About 20 minutes later, the agent is now on the outbnd2 job and has received notification that the Agent Owned Recall set previously is now coming due. |
| 21-24 | The agent continues on the outbnd2 job.   |

```
24 Client < 11:20:41 AGTReadyNextItem RAgent server 2035 300 2
0
M00000
25 Client < 11:29:29 AGTJobTransLink NAgent server 2035 0 3
0
M00001
outbnd1
26 Client < 11:29:29 AGTJobTransLink NAgent server 2035 0 2
0
M00000
27 Client > 11:29:35 AGTDetachJob C100 200 300 0
28 Client < 11:29:35 AGTDetachJob RAgent server 2035 300 2
0
M00000
29 Client > 11:29:57 AGTAttachJob C102 203 304 1
outbnd1
30 Client < 11:29:57 AGTAttachJob RAgent server 2035 304 2
0
```

## Avaya Proactive Contact Agent API 5.2

M00000

```
-----
31 Client > 11:30:00 AGTAvailWork C100 200 300 0
32 Client < 11:30:00 AGTAvailWork PAgent server 2035 300 2
0
S28833
33 Client < 11:30:00 AGTAvailWork RAgent server 2035 300 2
0
M00000
34 Client > 11:30:05 AGTReadyNextItem C100 200 300 0
35 Client < 11:30:05 AGTReadyNextItem RAgent server 2035 300 2
0
M00000
36 Client < 11:30:24 AGTCallNotify NAgent server 2035 0 4
0
M00001
Agent 04 Owned Recall
OUTBOUND
37 Client < 11:30:25 AGTCallNotify NAgent server 2035 0 2
0
M00000
38 Client > 11:31:02 AGTFinishedItem C100 200 300 1
23
```

Line	Meaning
25-26	About 10 minutes later, the agent application is notified that the agent must transfer to the outbnd1 job.
27-30	The agent application transfers the agent to outbnd1. Notice that the job setup notify key field and set data field are omitted.
31-35	The agent is now ready to start receiving calls on this job.
36-37	The agent application is notified of an Agent Owned Recall and the agent is sent the customer record and the outbound call.
38-41	The agent is finished with the customer record.

```
39 Client < 11:31:02 AGTFinishedItem RAgent server 2035 300 2
0
M00000
40 Client > 11:31:07 AGTReadyNextItem C100 200 300 0
41 Client < 11:31:07 AGTReadyNextItem RAgent server 2035 300 2
0
M00000
42 Client < 11:31:08 AGTJobTransLink NAgent server 2035 0 3
0
M00001
outbnd2
43 Client < 11:31:08 AGTJobTransLink NAgent server 2035 0 2
0
M00000
44 Client > 11:31:14 AGTDetachJob C100 200 300 0
45 Client < 11:31:14 AGTDetachJob RAgent server 2035 300 2
```

## Avaya Proactive Contact Agent API 5.2

```
0
M00000
46 Client > 11:31:33 AGTAttachJob C102 203 304 1
outbnd2
47 Client < 11:31:33 AGTAttachJob RAgent server 2035 304 2
0
M00000
48 Client > 11:31:38 AGTAvailWork C100 200 300 0
49 Client < 11:31:38 AGTAvailWork PAgent server 2035 300 2
0
S28833
50 Client < 11:31:39 AGTAvailWork RAgent server 2035 300 2
0
M00000
51 Client > 11:31:42 AGTReadyNextItem C100 200 300 0
52 Client < 11:31:42 AGTReadyNextItem RAgent server 2035 300 2
0
M00000
```

Line	Meaning
38-41	The agent is finished with the customer record.
42-43	The agent application is notified that the agent is now ready to go back to the outbnd2 job.
44-47	The agent application detaches from the outbnd1 job and attaches to the outbnd2 job.
48-52	The agent is now ready to start receiving calls again on outbnd2.

## Using Proactive Contact Agent API DLL

The Proactive Contact Agent API (Moagent32.dll) is a COM-compliant DLL, which provides the interfaces and dialog boxes that Windows application developers can use when creating agent applications. Agent applications developed with Moagent32.dll run on PCs meeting the standard Proactive Contact network-connected agent workstation configuration.

The Avaya Proactive Contact 4.0 and later agent server communicates over SSL. With Moagent.dll (Agent API) Avaya will supply the following additional binaries along with API:

- Third party ocx (WEONLYDO's TelnetDLX).
- Proactive Contact CA certificate.
- Agent certificate signed by the above CA certificate.
- Modified moagent.ini

---

Avaya Proactive Contact implements SSL (Secured Socket Layer) for communication starting from Version 4.0. Therefore, all the existing applications created using Avaya Proactive Contact 3.0 Agent API will need to be recompiled using Avaya Proactive Contact 5.1 Agent API.

This section contains the following topics:

- [Developer requirement](#)
- [The Proactive Contact Agent API and COM](#)
- [Work with the Moagent32.dll](#)
- [Use the interface methods](#)
- [Handle notification events from Proactive Contact](#)
- [Set default properties](#)
- [Use Moagent32.dll dialog boxes](#)
- [The Moagent32 Interface](#)
- [Method Syntax](#)
- [Moagent32.ini Settings](#)

## Developer requirements

The following are skills required of developers.

- Windows platform programming skills.
- Knowledge of Proactive Contact operation, interfaces, and methods.
- Knowledge of third-party development tools.
- Knowledge of COM (Component Object Model) architecture.

## The Proactive Contact Agent API and COM

Component Object Model (COM) is a general architecture for component software that defines a binary standard for method calling between components and provides for strongly-typed groupings of methods into “interfaces.”

COM also defines a base interface consisting of a group of standard methods that allows components to dynamically discover the interfaces implemented by other components and then manage interactions among components.

In general, you can develop COM components in any programming language with any tool, and they can run across networks from different machines. Avaya Communication has developed the Agent API to be compatible with applications developed using languages such as Visual Basic, PowerBuilder, and Visual C++. For more information on developing COM-compliant applications, review the COM information on Microsoft's web site.

Because the Proactive Contact Agent API is COM-compliant, all methods associated with the API are exported as the IMoagent interface that a client can access with minimal code. This chapter contains code examples for some commonly used languages.

The client application communicates with Proactive Contact by specifying an object name defined within the client, associating the object with the Proactive Contact Agent DLL interface object (class), and then accessing the required method using the “dot prefix technique”.

For example, a client might use the following syntax to call the “AttachJob” method:

```
call name.AttachJob(JobName, ErrCode, ErrText)
```

where **name** is the name of the client object that the client associates with the DLL interface object.

## Work with the Moagent32.dll

This section describes how to integrate your agent application with the Moagent32.dll. Topics covered include connecting to the DLL, using methods, setting default properties, and handling events from Proactive Contact. Code samples for Visual Basic, PowerBuilder, and Visual C++ are provided.

**NOTE:** COM dll Moagent32 is built as 32-bit. On Windows 64 bit OS, the Moagent32.dll was tested using sample client application developed in VC9 with Win32 settings(32-bit sample application).

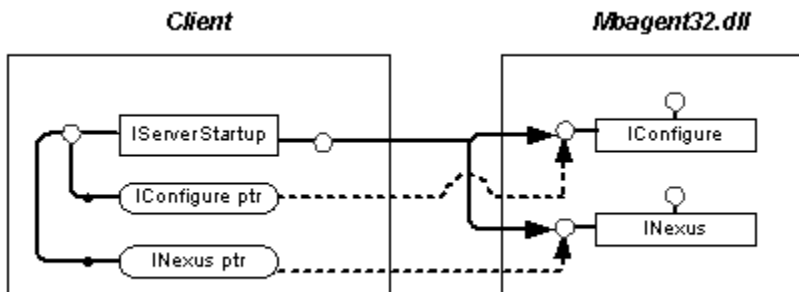
## Connect to the Moagent32.dll

To establish a connection between your client application and the Moagent32.dll, your application must create an object, "IServerStartup," to access the object (named "IMoagent") created and exported by the Moagent32.dll.

---

Your client application must use the object exported by Moagent32.dll. You cannot create an object of the same type ("IMoagent") from your client application.

The IServerStartup object contains a property that returns a pointer to the object ("IMoagent"). Make sure that your client application typecasts variables of type "IMoagent" and "IConfigure" and then associates these variables to the objects on Moagent32.dll (IConfigure and IMoagent) through your client's IServerStartup object. You can associate the variables to the object from your client application by setting the "IMoagent" variable to the ServerStartup property "PropIMoagent" and the "IConfigure" variable to "PropIConfig." The following diagram illustrates this process.



The following code samples demonstrate how to connect to the Moagent32.dll.

---

Code text appearing in *italics* indicates arbitrary names for the purposes of the code sample. You should replace italic text with the appropriate names from your client application.

### Visual Basic 5.0 and later

```

Public With Events MosServer As IMoagent 'COM server's interface class. Cannot
be created by the client.
Private SerStart As IServerStartup ' Establishes entry into server to get a
pointer to IMoagent (because IMoagent cannot be created by the client).
Private DllConfig As IConfigure ' Configures DLL settings.

' Create Objects
Set SerStart = New IServerStartup ' Entry point into DLL.
Set MosServer = SerStart.PropIMoagent ' Set MosServer to point to server object
IMoagent.
Set DllConfig = SerStart.PropIConfig ' Set DllConfig to point to server object
IConfigure.
  
```

### Power Builder 6.5 or later

```

// Declare global OLE automation objects.
OLEObject MosServer // COM object types
OLEObject MosSerStartup
OLEObject MosConfig
  
```

## Avaya Proactive Contact Agent API 5.2

```
// Create ActiveX object. OLEObjects are declared global.
MosServer = create OLEObject // Create Agent interface.
MosSerStartup = create OLEObject
MosConfig = create OLEObject
retval = MosSerStartup.ConnectToNewObject("Moagent32.IServerStartup")
if(0 <> retval ) then destroy MosServer messagebox("Server Error","Server
connect failed" + " " +
string(retval))
else // Successful connect to Agent Server.
MosServer = MosSerStartup.PropIMoagent
MosConfig = MosSerStartup.PropIConfig
MosSerStartup.PropWinHwnd = handle(mainfrm) //
MosSerStartup.PropWinUserMsg = 1035 // Set WM_USER for call

//notify.
end if
```

## Visual C++ 5.0 and later

```
// Creates the resource IDs and interface mappings
HRESULT GHr
= OleInitialize(NULL);
_IServerStartupPtr GptrServerStartup = NULL;
_IMoagentPtr GptrMoagent = NULL;
_IConfigurePtr GptrConfigure = NULL;
#import "moagent32.tlb" no_namespace

// Establish COM link to Moagent32.dll.
BOOL GetCOMMoagent32Ptr(void)
{
if(SUCCEEDED(CoInitialize(NULL))) // Initialize COM 'stuff'.
{
if(SUCCEEDED(GHr =
GptrServerStartup.CreateInstance(CLSID_IServerStartup)))
{
GptrMoagent = GptrServerStartup->PropIMoagent;

//IMoagent interface. All methods ( Proactive Contact commands) available
//through this pointer.
GptrConfigure = GptrServerStartup->PropIConfig;

//IConfigure interface. All configuration properties available
//through this pointer.
return 1;
}
}
return 0;
```

## Use the interface methods

The methods available within the IMoagent interface issue all Proactive Contact API commands. For example, commands used to log in to Proactive Contact or transfer a caller. These commands have the same name as the name of the method. A method is available for each Proactive Contact API command. All input arguments are strings and each method returns a Boolean type only when Proactive Contact completes the issued command. That is, all Moagent32 methods are synchronous.

To issue a Proactive Contact command from your client application, you have to call a method.

See [The Moagent32 Interface](#) the section later in this chapter for sample calls. See [Appendix D: Moagent32.dll Interfaces](#) on page 280 for a list of methods.

The following code samples demonstrate how to call a method.

### Visual Basic 5.0 and later

```
Private Sub ButArray_cmds_Click(Index As Integer)
Dim ErrCode As String
Dim ErrText As String
Dim DataBuf As String
Dim LstrCmd As String
Select Case Index
Case 0
If (Not MosServer.AttachJob("",ErrCode, ErrText))
Then GoTo ErrCond
Lbl_state(1).Caption = "On Job " & ErrText
LstrCmd = "AttachJob"
Call InstallDataFields
Case 1
If (Not MosServer.AvailWork(ErrCode, ErrText)) Then
GoTo ErrCond
Lbl_state(3).Caption = "Available for Work"
LstrCmd = "AvailWork"
Case 2
.
.
.
End Select
Err Cond:
Call ErrorCondition(ErrCode, ErrText)
End Sub
```

### Power Builder 6.5 and later

```
If (Not MosServer.AttachJob("SomeJobName",ref GstrErrCode,
ref GstrErrText)) Then
errorcondition(GstrErrCode,GstrErrText)
else
displayresults("AttachJob() successful. Attached to job
","")
end if
If (Not
MosServer.Logon(ServerName,PortNumber,GstrUid,GstrPwd,&
GstrHeadset,REF GstrErrCode,REF GstrErrText)) Then
errorcondition(GstrErrCode,GstrErrText)
else
displayresults("AttachJob() successful. Attached to job&
","")
end if
```

### Visual C++ 5.0 and later

```
//Method:Agent Attach Job
//.
.
case IDC_BUT_ATTACHJOB: // Call attach job method.
```



```
SUCCEEDED(GptrMoagent ->AttachJob(ToMos, &ErrCode, &ErrText)) ?  
ProcessError(ErrCode, ErrText, hDlg, MAINFRM_ERRS) :  
ProcessResponse(ErrCode, ErrText, hDlg, MAINFRM_MOSRESP, "AttachJob", FALSE, FromMos);  
break;  
  
//Method:Agent Deattach Job  
case IDC_BUTTON_DETACHJOB: // Call detach job method.  
SUCCEEDED(GptrMoagent->DetachJob(&ErrCode, &ErrText)) ?  
ProcessError(ErrCode, ErrText, hDlg, MAINFRM_ERRS) :  
ProcessResponse(ErrCode, ErrText, hDlg, MAINFRM_MOSRESP, "DetachJob", FALSE, FromMos);  
break;
```

## Handle notification events from Proactive Contact

Proactive Contact sends messages, “notifications”, to your client application in response to events that occur on Proactive Contact. Notifications can either be solicited by your client application or sent unsolicited by Proactive Contact. For example, if your client application issues the command associated with the ReadyNextItem method, Proactive Contact returns the CallNotify notification which provides the customer data. HeadsetConnBroken is an example of an unsolicited notify sent by Proactive Contact when an agent’s headset connection has been lost.

The notification events available with Moagent32.dll include:

- CallNotify
- HeadsetConnBroken
- licbAbort
- licbFeNotify
- licbOffline
- licbOnline
- JobEnd
- UnitEnd
- JobTransLink
- JobTransRequest
- JobMode
- PreviewRecord
- ReceiveMessage
- Start
- SystemError
- XferCustHangup
- XferTrunkHanup

For a description of each notification event, see [Commands and Notification Events](#). Some notification events are specific to the Moagent32.dll; data is not sent from Proactive Contact. See [Appendix D: Moagent32.dll Interfaces](#) on page 280 for descriptions of these events.

The following code samples demonstrate how you would handle Proactive Contact notification events in your client application.

### Visual Basic 5.0 and later

```
Private Sub MosServer_AvayaMosaixEvent(ErrFlag As Boolean,
NotifyType As String, AvayaMosaixDataPacket As String, ErrCode As
String, ErrText As String)
.
.
Client Code
.
.
End Sub
```

### Power Builder 6.5 and later

```
// This is event 'servernotify' with ID pbm_custom12 located in 'mainfrm' of
sample code.
// Received call notify from Proactive Contact via WM_USER = 1035 as set in
open event.
string NotifyType
string AvayaMosaixData
MosServer.GetCallNotify(ref NotifyType,ref AvayaMosaixData)
ProcessNotify(NotifyType,AvayaMosaixData) // Function to process Proactive
Contact
//call data.
```

### Visual C++ 5.0 and later

```
// CMoAgentEvent
//
// Description: This class blends ATL with the project (MFC) and enables use of
// the ATL macros and templates for handling a dispinterface and events.
//
class ATL_NO_VTABLE CMoAgentEvent :
.
.
.
// This template is tied to the SINK_MAP macro
// (UniqueId - user choice, this class, Id for __IMoagent, Type Library,
// version, version indice)
public IDispatchImpl<42, CMoAgentEvent, &DIID__IMoagent, &LIBID_Moagent32, 1,
0>
...
...

// Local function for handling the event from __IMoAgent.
// This function needs to map to the method on the interface.
void _stdcall OnMoAgentData(VARIANT_BOOL* err_flag, BSTR*
notify, \BSTR* moagent_data, BSTR* err_code, BSTR*
err_text);
...
...
```

```
// The following macro mapping is used to tie events from interfaces to
// a local function for handling the events.
BEGIN_SINK_MAP(CMoAgentEvent)
// (UniqueId - user choice, Id for __IMoagent, Dispid, your local function)
SINK_ENTRY_EX(42, DIID__IMoagent, 0x1, OnMoAgentData)
END_SINK_MAP()

.
.
.
.// Method:OnMoAgentData
//
// Description: Handles the information sent from the event raised from
// the method AvayaMosaixEvent within the __IMoagent interface.
//
void STDMETHODCALLTYPE CMoAgentEvent::OnMoAgentData(VARIANT_BOOL*
err_flag, BSTR* notify, BSTR* moagent_data, BSTR* err_code,
BSTR* err_text)
{
    CString data_str = *moagent_data;
    CString notify_str = *notify;
    CString err_code_str = *err_code;
    CString err_text_str = *err_text;

    // Add your parsing of the data here
}
```

## Set default properties

The default properties for the Moagent32.dll (IConfigure Interface) are listed in [Appendix D: Moagent32.dll Interfaces](#) on page 280. You can specify a different property value from your client application.

The default SetCreateLogFile property is **True**. You can program an option in the agent application to turn the property **on** and **off**. The Moagent32.log file is required to receive technical support from your Proactive Contact representative.

## Code samples

The following code samples demonstrate how to set default properties. Code text appearing in *italics* indicates arbitrary names for the purposes of the code sample. You should replace italic text with the appropriate names from your client application.

## Visual Basic 5.0 and later

```
Rem Configure Moagent32.dll
DllConfig.SetUseDllDbs = True 'Makes several standard dialog boxes available to
the client.
DllConfig.SetCreateLogFile = True 'Create log file of all AvayaMosaix-
Moagent32.dll
transactions.
DllConfig.SetLogFileName = "c:\moslog\moagent32.log"
DllConfig.SetCreateErrFile = False 'Write all errors (AvayaMosaix, DLL,
client) to log file.
DllConfig.SetErrFileName = "c:\moslog\moagent32.err"
DllConfig.SetNumOfLstErrs = 5 'Set the number of errors to track during a
client session. Use the method DllConfig.DisplayLastErrors to view a errors
```

## Avaya Proactive Contact Agent API 5.2

```
list. DllConfig.SetLogonRecovery = True 'If client submits invalid login
parameters (user ID, passwd), DLL will post a dialog box requesting valid
login data.
DllConfig.SetLogonTimeout = 20 'If the agent does not respond to an issued
command within the specified interval (seconds), DLL times out and returns
control to client.
DllConfig.SetReserveConnHeadSet = True 'Allows DllConfig.Logon to reserve and
connect the agent headset automatically upon a successful login.
DllConfig.SetAvayaMosaixTimeout = 20 'If Proactive Contact does not respond to
an issued command within the specified interval (seconds), DLL times out and
returns control to client.
```

### Power Builder 6.5 or later

```
// Set properties
MosConfig.SetUseDllDbs = True // Makes several standard dialog boxes

// available to the client.
MosConfig.SetCreateLogFile = True // Create log file of all Proactive Contact-

//Moagent32.dll transactions.
MosConfig.SetLogFileName = "c:\moslog\moagent32.log"
MosConfig.SetCreateErrFile = False // Write all errors (AvayaMosaix, DLL,

// client) to log file.
MosConfig.SetErrFileName = "c:\moslog\moagent32.err"
MosConfig.SetNumOfLstErrs = 5 // Set the number of errors to track

// during a client session. Use the method MosConfig.DisplayLastErrors to view
// an errors list.
MosConfig.SetLogonRecovery = True // If client submits invalid login

// parameters (user ID, passwd), DLL will post a dialog box requesting valid
// logon data.
MosConfig.SetLogonTimeout = 20 // If the agent does not respond to an

// issued command within the specified interval (seconds), DLL times out and
// returns control to client.
MosConfig.SetReserveConnHeadSet = True // Allows MosConfig.Logon

// to reserve and connect the agent headset automatically upon a successful
logon.
MosConfig.SetAvayaMosaixTimeout = 20 // If Proactive Contact does not respond

// to an issued command within the specified interval (seconds), DLL times out
// and returns control to client.
```

### Visual Basic C++ 5.0 and later

```
VARIANT_BOOL flag_val;

//get the pointer for the _IConfigure interface
m_pServer->get_PropIConfig(&m_pConfig);

//retrieve reserve headset value
flag_val = m_pConfig->GetResConnHead;
flag_val = VB_TRUE; //Set for VB_TRUE (-1)

//set the reserve headset value
m_pConfig->PutSetReserveConnHeadset(&flag_val);
```

## Avaya Proactive Contact Agent API 5.2

```
// make several standard dialog
//boxes available to the client.
m_pConfig->PutSetUseDllDlls = True

// create log file of all AvayaMosaix-Moagent32.dll transactions
m_pConfig->PutSetCreateLogFile = True

// name the log file moagent32.log
m_pConfig->PutSetLogFileName = "c:\moslog\moagent32.log"

// write all errors (AvayaMosaix, DLL, client) to log file
m_pConfig->PutSetCreateErrFile = False

// name the error file moagent32.err
m_pConfig->PutSetErrFileName = "c:\moslog\moagent32.err"

// set the number of errors to track during a client session to 5. Use the
method
//MosConfig.DisplayLastErrors to view an errors list
m_pConfig->PutSetNumOfLstErrs = 5

// when client submits invalid logon parameters (user ID, passwd), the DLL
will
//post a dialog box requesting valid logon data
m_pConfig->PutSetLogonRecovery = True

// when the agent does not respond to an issued command within 20 seconds,
//the DLL will time out and return control to the client
m_pConfig->PutSetLogonTimeout = 20

// reserves and connects the agent headset automatically upon a successful
logon
m_pConfig->PutSetReserveConnHeadSet = True

// when Proactive Contact does not respond to an issued command within the
//20 seconds, the DLL times out and returns control to the client
m_pConfig->PutSetAvayaMosaixTimeout = 20
```

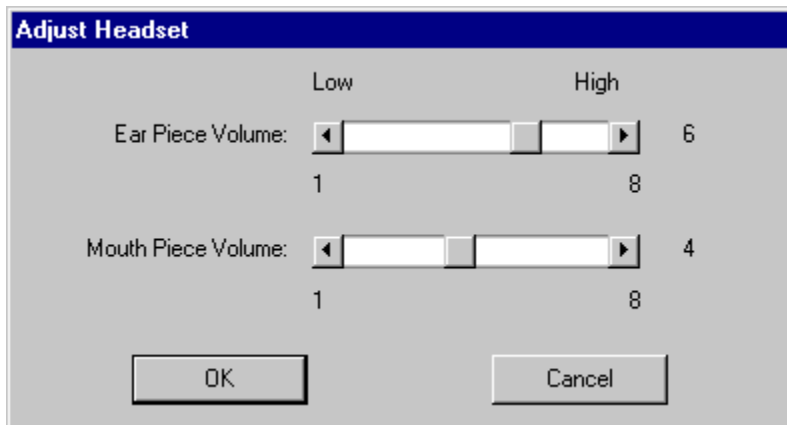
## Use Moagent32.dll dialog boxes

Moagent32.dll provides a set of dialog boxes. You can choose to use these dialog boxes in your client application. You invoke a dialog box from your application by submitting the associated command with empty parameters. The IConfigure interface property and the SetUseDllDlls need to be set to **True** (default value) to use these dialog boxes.

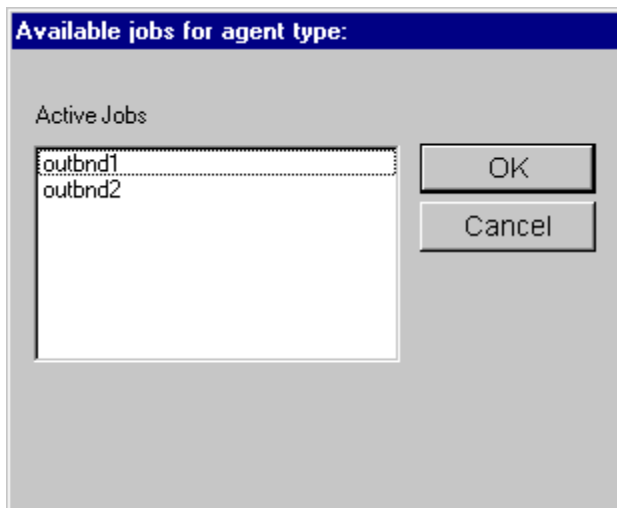
If you choose not to use the dialog boxes, set SetUseDllDlls to **False** and provide the appropriate arguments in each command.

The following dialog boxes are a sample of those available. Each illustrates how to invoke the dialog box.

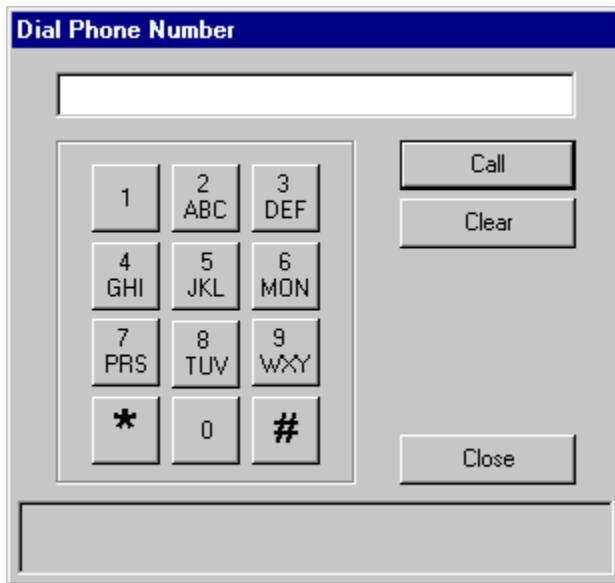
AdjustHeadset ("","",ErrCode,ErrText)



AttachJob ("",ErrCode,ErrText)

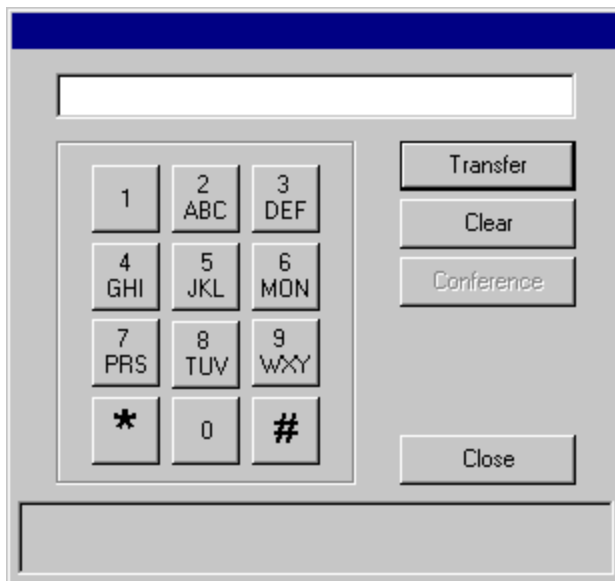


ManualCall ("",ErrCode,ErrText)



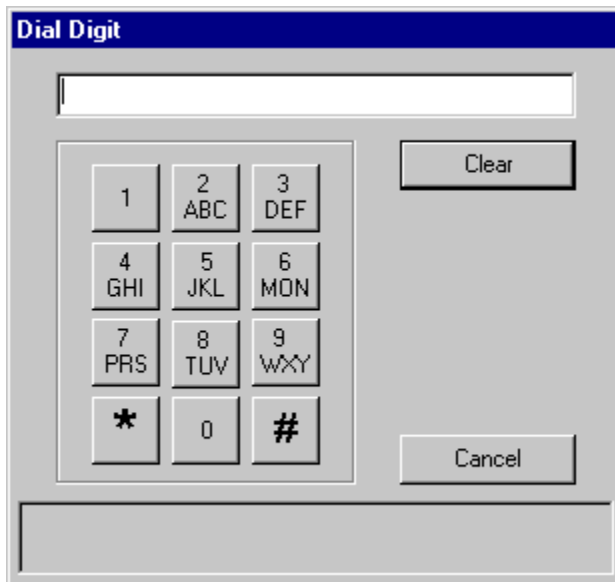
The 'Dial Phone Number' dialog box features a blue title bar. Below it is a white text input field. A numeric keypad is positioned on the left, with buttons for digits 1-9, \*, 0, and #, each accompanied by its respective letters (e.g., 2 has ABC, 3 has DEF). To the right of the keypad are three buttons: 'Call', 'Clear', and 'Close'. A wide, empty rectangular area is located at the bottom of the dialog.

TransferCall ("",ErrCode,ErrText)



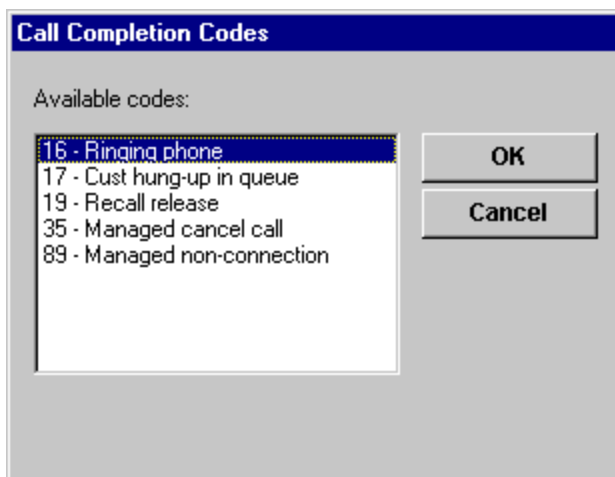
The 'TransferCall' dialog box has a blue title bar and a white text input field. It includes a numeric keypad on the left with buttons for digits 1-9, \*, 0, and #, each with its corresponding letters. To the right of the keypad are four buttons: 'Transfer', 'Clear', 'Conference', and 'Close'. A wide, empty rectangular area is at the bottom of the dialog.

DialDigit ("",ErrCode,ErrText)



The 'Dial Digit' dialog box features a blue title bar. Below it is a text input field. A numeric keypad is positioned in the center-left, with buttons for digits 1-9, \*, 0, and #. Each digit button also includes its corresponding letters (e.g., 2 has ABC, 3 has DEF). To the right of the keypad are 'Clear' and 'Cancel' buttons. A status bar is located at the bottom of the dialog.

FinishedItem ("",ErrCode,ErrText)

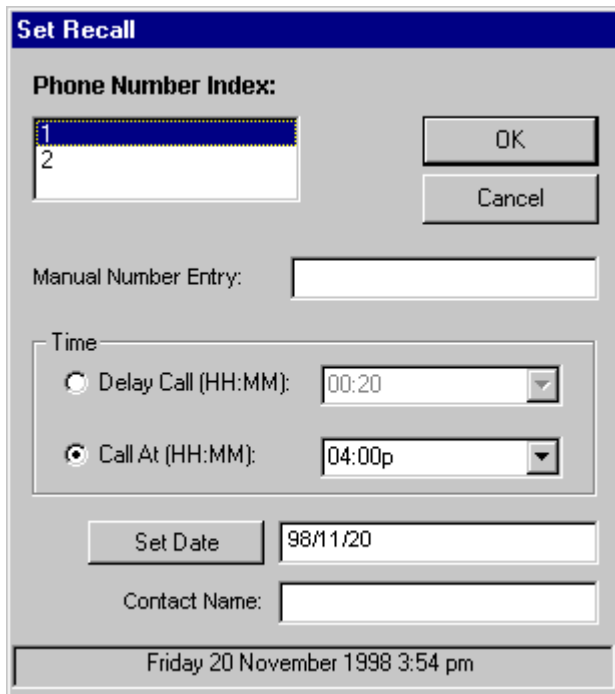


The 'Call Completion Codes' dialog box has a blue title bar. It contains a label 'Available codes:' followed by a list box. The list box contains five items: '16 - Ringing phone', '17 - Cust hung-up in queue', '19 - Recall release', '35 - Managed cancel call', and '89 - Managed non-connection'. The first item is selected. To the right of the list box are 'OK' and 'Cancel' buttons.

Code	Description
16	Ringing phone
17	Cust hung-up in queue
19	Recall release
35	Managed cancel call
89	Managed non-connection



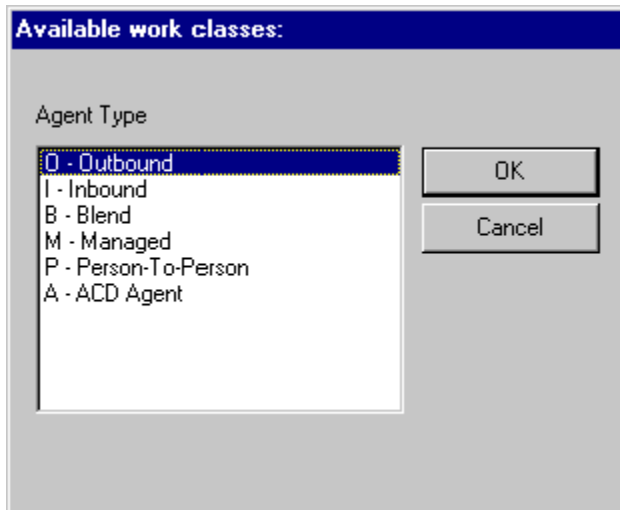
SetCallback ("", "", "", "", "", ErrCode, ErrText)



The 'Set Recall' dialog box has a blue title bar. It contains a 'Phone Number Index' section with a list box showing '1' and '2', and 'OK' and 'Cancel' buttons. Below this is a 'Manual Number Entry' text field. A 'Time' section contains two radio buttons: 'Delay Call (HH:MM):' with a dropdown set to '00:20', and 'Call At (HH:MM):' with a dropdown set to '04:00p'. Below the time section is a 'Set Date' button and a date field showing '98/11/20'. At the bottom is a 'Contact Name' text field. A status bar at the very bottom displays 'Friday 20 November 1998 3:54 pm'.

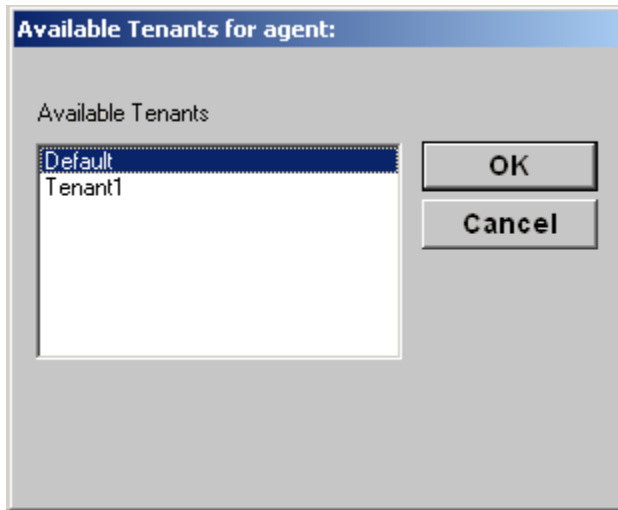
Proactive Contact uses the 10-character date format with 4-digit years, for example 2002/11/20.

SetWorkClass ("", ErrCode, ErrText)



The 'Available work classes' dialog box has a blue title bar. It features an 'Agent Type' section with a list box containing 'O - Outbound', 'I - Inbound', 'B - Blend', 'M - Managed', 'P - Person-To-Person', and 'A - ACD Agent'. To the right of the list box are 'OK' and 'Cancel' buttons.

SetTenant("", ErrCode, ErrText)



## The Moagent32 Interface

The methods (functions) that make up the Moagent32 interface (IMoagent) consist of four distinct groups:

Methods with no Proactive Contact arguments that do not return data

Methods with no Proactive Contact arguments that return data

Methods with arguments that do not return data

Methods with arguments that return data

The methods available with the Moagent32.dll issue Proactive Contact API commands that have the same name as the name of the method. See [Appendix D: Moagent32.dll Interfaces](#) on page 280 for a list of methods and their arguments.

---

Some methods are specific to the Moagent32.dll (that is, returned data does not come from Proactive Contact).

## Method Syntax

In the following examples, name refers to the name assigned by the agent application to the Moagent32 object.

---

Ensure that you allocate adequate memory (about 100K) in your client application to accommodate the ErrCode and ErrText strings.

The following code shows sample Method syntax:

```
Methods with no Avaya Proactive Contact arguments that do not return data (no
args, no data)
Called as: name.MethodName(ErrCode as string, ErrText as string)
Methods with no Avaya Proactive Contact arguments that return data (no args,
data)
Called as: name.MethodName(ReturnData as string, ErrCode as string, ErrText as
string)
```

Methods with arguments that do not return data (args, no data)  
Called as: `MethodName(Arg1 as string, Arg2 as string, ErrCode as string, ErrText as string)`  
Methods with arguments that return data (args, data)  
Called as: `name.MethodName(Arg1 as string, Arg2 as string, RetData as string, ErrCode as string, ErrText as string)`

---

You can use string variables or literals.

If you use a method that does not require any arguments, the parentheses following the command name still need to be present (but are left empty). For example, `ConnHeadset()`.

## Moagent32.ini Settings

Use the `Moagent32.ini` file to localize (or customize) the message and dialog box text associated with the `Moagent32.dll`. The suggested location for the `Moagent32.ini` file is your project directory.

The following sections describe the parts of the `Moagent32.ini` file you can modify.

### [Logon]

Contains the login parameters for establishing a session with Proactive Contact. Do not change this section during localization. Although the login process requires the user name and password, they are not included in `Moagent32.ini`. `Moagent32.dll` uses a login dialog box to gather the data.

The logon section requires the following parameters:

Servername	Proactive Contact name
Portnumber	Proactive Contact port number. Set to 22700. Use this setting. If changed, the DLL will not connect to Proactive Contact.
Headset ID	Optional. If not included, <code>Moagent32.dll</code> will prompt for an ID during the login process.

### [Security]

The communications from and to the Avaya Proactive Contact agent is secured at the highest security level. The traditional TCP socket communication is changed to TCP over SSL (Secure Socket Layer).

The security settings require the following parameters:

`ServerCertificate=TRUE/FALSE`

`ClientCertificate=FALSE/TRUE`

`ClientCertificatePath=<certificate file path>`

ClientPrivateKeyPath=<Private key file path>

SSLMethod=SSLv23/TLSv1/TLSv11/TLSv12

By default SSLMethod = SSLv23 is set.

The default value of SSLMethod, “SSLv23”, in the Moagent32.ini file allows the communication with to dialer to happen using SSLv23/TLSv1/TLSv1.1/TLSv1.2 depending on the setting on the dialer server. For example, if Dialer is configured to use TLSv1.2, then the agent communication with PC 5.2 dialer will happen over TLSv1.2.

### [Work\_class]

Contains all available agent type codes. While agents can use any of these codes, individual Proactive Contact installations might not use some agent types, such as Managed or Person to Person.

Changing agent types in this list does not change the available agent types on Proactive Contact. It might, however, make existing agent types unavailable.

The code entries appear as numbered items in Moagent32.ini. For example, the first line of the section reads:

1= O - Outbound

For localization, you can translate the descriptions (such as “Outbound”).

However, do not change the numbering or the letter codes.

### [Dialog\_box\_text]

Contains the text strings that appear in the agent application dialog boxes. Each dialog box included in Moagent32.dll has one or more settings in this section. Changing these settings customizes the dialog boxes.

The Login box section contains two lines used by all dialog boxes: LogonOkBut and LogonCanBut. These lines contain the values that appear on all **OK** and **Cancel** buttons.

For localization purposes, translate the dialog box text into a language that agents using the agent application will understand.

If you add additional dialog boxes for your application, add the values for each dialog box to this section.

### [Agent\_state]

Contains the text messages that describe the various agent states. You can display these messages from your agent application when you call the ListState method.

For localization purposes, translate the message text into a language that agents using the agent application will understand.

### **[Server\_return\_codes]**

Contains explanatory text for the various Proactive Contact error codes that might return in response to Agent API commands. The error codes are part of the Proactive Contact binary files and cannot be changed.

Changing any of the text in this section can create false interpretations of Proactive Contact error codes. For localization, translate **ONLY** the text portion of these messages.

Do not change the Proactive Contact error codes that appear as part of the message text.

### **[Winsockerrors]**

Contains the text for wi nsock.ocx errors. Do not change message numbers.

### **[development\_platform]**

Contains the flag “.NET”. Default value of this flag is 0. If the client application is developed using C# and VB.NET, then set this flag to 1.

## Commands and Notification Events

This section contains the commands and notification events that make up the Proactive Contact Agent API. They are the methods used for IMoagent. See [Appendix D: Moagent32.dll Interfaces](#) for information on IServerStartUp and IConfigure.

Commands and events fall into one of five categories: connection commands, system commands, job setup commands, working commands, and notification events.

---

Each command and notification event appears in two formats: with the AGT prefix and without the prefix. This chapter lists each with the AGT prefix. Use the command or notification event without the prefix when creating an agent application. The Agent API and agent binary append the prefix AGT to each command and notification event message.

In the command and notification event message format section, the data segments appear on one line. In the log, each data segment is on a separate line.

In the examples, the <Client> parameter COriginator\_ID represents the user ID that originated the request. The parameter Agent server represents the Proactive Contact system ID. For general information regarding command and event types, see [Understanding the Proactive Contact Agent API](#).

### AGTAdjustHeadset

AGTAdjustHeadset is a working command.

#### Alternate name

AdjustHeadset

#### Function

Changes the volume settings for the headset ear or mouth piece. (Use AGTGetHeadsetVol to get the current volume settings.) Releasing the telephone line disables this command.

#### Availability

The command is available when the agent is working with a customer record on an open telephone line.

This setting is only available for direct connect headsets on Proactive Contact systems using OLIC cards.

Agents can use the PBX settings to adjust volume for headsets that are not direct connect.

---

An agent gets an open telephone line by receiving a call from a customer. To hang up a call with a customer and keep the line open, execute AGTHangupCall or AGTManualCall.

This command is not available on an Avaya Proactive Contact with CTI system.

**Format**

AGTAdjustHeadset C <Client> <ProcID> <InvokeID> 2§

<EarMouth>§ <Volume>‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

EarMouth	The setting to change: E (ear) or M (mouth).
Volume	The numeric (1-8) value for the volume setting. Use whole numbers only, no decimal values.

**Return values**

Code	Description
E28850	Cannot open the channel to the operator monitor process. Indicates an Proactive Contact system problem.
E28866	Telephone line is not available. The agent must receive a customer call to acquire an open telephone line.
E28867	Telephone line is not off-hook. The agent has an open telephone line that is on hold.
E28869	Headset volume must be in the range of one to eight. Retry the command using a number from one to eight in the <Volume> parameter.
E29950	This feature is not available on an Proactive Contact with CTI system.
M00000	Complete.
S28814	Transfer is in progress. The last call is being transferred. Retry the command after connecting to the next call.

## Examples

### Agent Application to Agent Binary

AGTAdjustHeadset C COriginator\_ID 11111 121 2§

M §

2‡

### Agent Binary to Agent Application

AGTAdjustHeadset R Agent server 29722 121 2§

0 §

M00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## See also

[AGTGetHeadsetVol](#)

[AGTHangupCall](#)

[AGTManualCall](#)

## AGTAORNotify

AGTAORNotify is a notification event message from the server.

### Alternate name

AORNotify

### Function

AORNotify is an unsolicited event message sent to an agent before he is transferred to a different job to take an Agent Owned Recall (AOR) call.

### Availability

This event occurs when an agent is logged in to a job and is ready for the next customer call.

### Format

AGTAORNotify NAgent server <ProclD> <InvokeID> 6 RS 0 RS M00001 RS  
CustomerName RS XferJobName RS UnitId RS OrigJobName

Where RS is the record separator.



**Data parameter**

CustomerName	The name of the customer that is being recalled.
XferJobName	The name of the job the agent is being transferred to for the recall.
UnitId	The unit work list value (if applicable) the agent specified for the original call.
OrigJobName	The name of the job where the recall was specified.

**Return values**

Code	Description
E28866	Telephone line is not available. The agent must receive a customer call to acquire an open telephone line.
E28867	Telephone line is not off-hook. The agent has an open telephone line that is on hold.
E28869	Headset volume must be in the range of one to eight. Retry the command using a number from one to eight in the <Volume> parameter.
M00000	Complete.
S28814	Transfer is in progress. The last call is being transferred. Retry the command after connecting to the next call.

**Examples**

```
AGTAORNotify  NAgent server  19917 0    6    -0-M00001-JOHN DOE-shadowjob_1-
Allid-outbnd
```

```
AGTAORNotify  NAgent server  19917 0    2    -0-M00000
```

## AGTAttachJob

AttachJob is a system command.

**Alternate Name**

AttachJob

**Function**

Attaches the agent application to a specific Proactive Contact job when the agent selects the job. This permits the agent application access to job-related information and job setup commands.

An agent application can attach to one job at a time. To change jobs, the agent clears the current job selection. This causes the agent application to detach the current job. The agent then selects a new job.

**Availability**

The agent application can only attach jobs that are active. Use AGTListJobs to see a list of jobs and their statuses.

This command is available any time after executing AGTLogon.

Execute this command before AGTAvailWork.

**Format**

AGTAttachJob C <Client> <ProcID> <InvokeID> 1§

<JobName>‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameter**

JobName	The value is the name of an active job on Proactive Contact. Use up to 19 characters, 7-bit USASCII. The <JobName> is case sensitive and might not include special characters or embedded spaces.
---------	---

**Return values**

Code	Description
E28804	Job <JobName> is not running (not active). This message usually indicates that a job is still starting.
E28889	The agent application is already attached to a job. The agent must clear the current job selection before selecting a new one.
E28890	Cannot open the job's resource file. This message usually indicates an Proactive Contact system problem or that a job is still starting.

Code	Description
E29206	Cannot attach the <Unit> segment of shared memory that Proactive Contact is using for the job. Most jobs use the same unit of shared memory, defined in a configuration file on the Proactive Contact system. This message indicates a problem with the Proactive Contact system configuration.
E58021	Tenant is not set, set tenant first. This message indicates that the tenant is not set. Tenant must be set before joining the job.
M00000	Complete.
S28833	Pending.

### Examples

#### Agent Application to Agent Binary

```
AGTAttachJob C COriginator_ID 11111 201 1$
managed‡
```

#### Agent Binary to Agent Application

```
AGTAttachJob R Agent server 5846 201 2$0 $
M00000‡
```

#### See also

[AGTAvailWork](#)

[AGTDetachJob](#)

[AGTListJobs](#)

[AGTLogon](#)

## AGTAutoReleaseLine

AGTAutoReleaseLine is a notification event that notifies when the phone line has been released.

#### Alternate name

AutoReleaseLine

**Function**

Sent when a customer hangs up the call if the job is set for Auto Update. When the AUTO RELEASE feature is turned on, the phone line is automatically released by the system if the customer hangs up the call first. When it happens, the corresponding phone line is released automatically without agent's effort. The talk timer is stopped as well.

**Availability**

This command is available on outbound capable jobs when the AUTO RELEASE feature is turned on and customer hangs up the call first.

**Format**

AGTAutoReleaseLine N <Agentserver> <ProcID> <InvokeID> 2§

0 § M00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None

**Return values**

Code	Description
M00000	Complete

**See also**

[AGTXferCustHangup](#)

[AGTXferTrunkHangup](#)

## AGTAutorelToReady

AGTAutorelToReady is a notification event that notifies client agent application to go into ready mode.

**Alternate name**

AutorelToReady

**Function**

Sent when the Answering Machine/FAX is detected and if the feature “Autorelease the agent into ready mode” feature is enabled in job. When this feature is configured, the system disconnects the call that is detected as Answering Machine or FAX. After receiving this event, agent application needs to send AGTReleaseLine and AGTFinishedItem (with the completion code received in AGTAutorelToReady)

**Availability**

This notification is available on outbound capable jobs when the “Autorelease the agent into ready mode” feature on.

**Format**

```
AGTAutorelToReady N Agentserver <ProcID> <InvokeID> 3$
0 § M00000 § <CompCode>‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

< CompCode >	The value is the completion code of Answering Machine or FAX.
--------------	---

**Return values**

Code	Description
M00000	Complete

```
AGTAutorelToReady N Agent server 4880 0 3$
0 §
M00001 §
15‡
AGTAutorelToReady N Agent server 4880 0 2$
0 §
M00000 §
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## AGTAvailWork

AGTAvailWork is a job setup command.

### Alternate name

AvailWork

### Function

Makes the agent available for work on the current job.

Use this command after an agent joins (logs on to) the job.

Execute this command anytime after AGTConnHeadset and AGTattachJob for non-ACD agents.

Execute this command everytime after AGTlicbOnline for ACD agents.

Execute AGTReadyNextItem when the agent is ready to receive a call from Proactive Contact.

To log the agent out of an active job, the agent application executes the AGTNoFurtherWork command.

Proactive Contact or the supervisor might also log the agent out of the job (see [AGTJobEnd](#), [AGTJobTransLink](#), [AGTJobTransRequest](#)).

### Availability

Execute this command any time after [AGTConnHeadset](#) and [AGTAttachJob](#) for non-ACD agents.

Execute this command every time after [AGTlicbOnline](#) for ACD agents.

### Format

AGTAvailWork C <Client> <ProcID> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

None.

### Return values

Code	Description

Code	Description
E28805	§ <JobName>‡ <JobName> is not ready. The attached job is active but is not placing calls. Retry the logon after job starts placing calls.
E28813	No more agents of this type can join the job. Job parameters include a maximum number of agents of each type (work class) that can log in to the job. See <a href="#">AGTSetWorkClass</a> .
E28814	Managed agents cannot join this job. The agent type (work class) is M (Managed). The attached job is not a Managed Dialing job. Reset the agent type and retry.
E28815	The agent is logged in as a Unit Work List agent and is attempting to join a Sales Verification job. Select a different job and retry.
E28816	Only Inbound agents can join this job. The agent application agent type (work class) does not match the job type. Set the agent type to I and retry.
E28817	Only Outbound agents can join this job. The agent application agent type (work class) does not match the job type. Set the agent type to O and retry.
E28818	Only Outbound or Managed agents can join this job. The agent application agent type (work class) does not match the job type. Set the agent type to O or M and retry.
E28819	Only Outbound agents can join Sales Verification jobs. The agent application agent type (work class) does not match the job type. Set the agent type to O and retry.
E28885	The agent application is not attached to a job. The agent must select a job before joining it.
E28895	The agent is already available for work.
E28896	The agent's headset must be active. Execute AGTConnHeadset and retry.
E28897	An available for work request is already pending.
E28898	The job is not available for logon. The job might have become inactive since it was attached. Execute AGTListJobs to check the job status.

Code	Description
E28899	There is no available for work request pending. AGTAvailWork did not execute. This message can indicate an Proactive Contact system problem.
E28900	§ <message>‡ An Proactive Contact system internal error occurred. Error message text follows.
E28946	Agent not yet acquired. The agent is logged in as an Agent Blending agent, but Proactive Contact has not acquired the agent for outbound calling.
E29000	Only Managed agents can join this job. The agent application agent type (work class) does not match the job type. Reset the agent type to M and retry.
E29952	Failed to join job. Please contact system administrator. This error code appears when there is a licensing issue. If the system is not able to contact the enforcer service.
E70006	Unit ID value not selected. The agent is logged in to a Unit Work List job but has not selected a Unit ID value to work with. Execute AGTSetUnit and retry.
M00000	Complete.
S28833	Pending.

## Examples

### Agent Application to Agent Binary

```
AGTAvailWork C COriginator_ID 11111 17 0‡
Agent Binary to Agent Application
AGTAvailWork P Agent server 17970 17 2§
0 §
S28833‡
AGTAvailWork R Agent server 17970 17 2§
0 §
M00000‡
```

## See also

[AGTAttachJob](#)

[AGTConnHeadset](#)

[AGTJobEnd](#)



[AGTJobTransLink](#)

[AGTJobTransRequest](#)

[AGTListJobs](#)

[AGTNoFurtherWork](#)

[AGTSetWorkClass](#)

## AGTCallNotify

AGTCallNotify is a notification event message from the server.

### Alternate Name

CallNotify

### Function

Notifies the agent application of a call that is coming to the agent.

The agent binary sends at least two messages to the agent application for each call notification event:

The first message contains basic information. It generally contains the caller's name, a text string relating to how long the customer waited, the type of call (inbound or outbound), the key field name, and key field value. If the call is a voice and data transfer, the message "TRANSFER CALL" appears instead of the customer's name. If the call is an agent owned recall, the agent binary notifies the agent of a pending recall at the increment set by the RECALL\_NOTIFY entry in the job file.

A second message contains more information about the customer. The agent binary does not send the second message if the agent application has not requested additional information by executing AGTSetDataField. If the agent or the agent application requests multiple fields, AGTCallNotify sends the field names and values in the order the AGTSetDataField commands executed.

The final message is always the M00000 completion record.

If the agent binary receives an unexpected call notification when the agent state is not appropriate to receive a call, error messages might replace the AGTCallNotify event messages.

### Availability

This event occurs when an agent is logged in to a job and is ready for the next customer call.

### Format

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

AGTCallNotify N <Agentserver> <ProcID> <InvokeID> 5§

0 § M00001 § <OpMesg> [<WaitMsg>] §

<CallType> §<NotifyFieldName>, <NotifyFieldData>‡

AGTCallNotify N <Agentserver> <ProcID> <InvokeID> <n

\*

> (Where n equals the number of segments.)§

0 §M00001 § <FieldName>, <FieldData> §

<FieldName>, <FieldData>...‡

### Data parameters

OpMesg	<p>OpMesg Message from call notification containing field information from the customer record configured in alljobs.dat on the Proactive Contact system. The default field is NAME.</p> <p>If the call is a voice and data transfer from another agent, the &lt;OpMesg&gt; is "TRANSFER CALL."</p> <p>If the call is an Agent Owned Recall, the agent binary notifies the agent of a pending recall at the increment set by the RECALL_NOTIFY entry in the job file.</p> <p>The pending notification is:</p> <p>"AOR: &lt;Customer Name&gt; J: &lt;job name&gt; U: &lt;user ID&gt; T: HH.MM.SS."</p> <p>In this case the &lt;OpMesg&gt; contains the field name present in Alljobs.dat and the pending notification separated by pipe( ).</p> <p>Example:</p> <p>Home phone - 2037538811   AOR: JOHN DOE Job: outbnd Unit: Allid Time: 16.50.00</p> <p>The &lt;OpMesg&gt; for the recall is "Agent &lt;Agent ID&gt; Owned Recall."</p>
WaitMsg	<p>If present, an asterisk (*) separates &lt;WaitMsg&gt; from &lt;OpMesg&gt;. &lt;WaitMsg&gt; is a text string defined in the Proactive Contact job file. It indicates how long the customer has been on hold.</p> <p>&lt;WaitMsg&gt; can be up to 30 characters long. The message can be for the agent to read to the customer ("We're sorry you had to hold."), or it can be for the agent's information (5-10 seconds).</p>

CallType	Identifies the call direction. If Proactive Contact places the call, it is OUTBOUND. If the customer places the call, it is INBOUND. An agent can own an outbound call placed during an Outbound, Blend, Managed Dialing, and Unit Work List job.
NotifyFieldName	The calling list field name requested by AGTSetNotifyKeyField.
NotifyFieldData	The notification key field value in the customer's record.
FieldName	A calling list field name requested by AGTSetDataField.
FieldData	The customer record field data.

**Return values**

Code	Description
E28885	The agent application is not attached to a job. The call notification event is inappropriate.
E28900	§ <Message>‡ An Proactive Contact system internal error occurred. Error message text follows.
E28906	The agent is not ready for next customer record. Call notification event is inappropriate.
M00000	Complete.
M00001	§ <OpMesg>[*<WaitMsg>] § <CallType> § <NotifyFieldName>,<NotifyFieldData>‡ Initial notification data message.
M00001	§ <FieldName>,<FieldData> § <FieldName>,<FieldData>...‡ Additional fields data message.

**Examples****Agent Binary to Agent Application**

```
AGTCallNotify N Agent server 6111 0 5$
0 §
M00001 §
```

## Avaya Proactive Contact Agent API 5.2

JOHN DOE \$  
OUTBOUND \$  
ACCTNUM, 4302209860039647+  
AGTCallNotify N Agent server 6111 0 3\$  
0 \$  
M00001 \$  
BAL, 4368+  
AGTCallNotify N Agent server 6111 0 2\$  
0 \$  
M00000+

### Agent Binary to Agent Application

AGTCallNotify N Agent server 6111 0 5\$  
0 \$  
M00001 \$  
ANGEL CORTEZ\*SORRY YOU HAD TO HOLD \$  
INBOUND \$  
ACCTNUM, 97431672348572947+  
AGTCallNotify N Agent server 6111 0 2  
0 \$  
M00000+  
AGTCallNotify N Agent server 6111 0 5  
0 \$  
M00001 \$  
TRANSFER CALL \$  
INBOUND \$  
ACCTNUM, 9724973513683608+  
AGTCallNotify N Agent server 6111 0 2\$  
0 \$  
M00000+  
AGTCallNotify N Agent server 6111 0 2\$  
0 \$  
M00001+  
Home phone - 2037538811 | AOR: JOHN DOE J: outbnd1 U: Allid T: 11.26.00 (Agent  
binary sends this message to notify Agent of the pending owned recall.)  
OUTBOUND  
ACCTNUM, 55555223331234567  
AGTCallNotify N Agent server 6111 0 2\$  
0 \$  
M00001+  
Agent agent4 Owned Recall (Agent binary sends this message when Agent 04  
receives the owned recall.)  
OUTBOUND  
ACCTNUM, 55555223331234568

### See also

[AGTFinishedItem](#)

[AGTReadyNextItem](#)

[AGTSetDataField](#)

[AGTSetNotifyKeyField](#)

## AGTCancelSearch

AGTCancelSearch is a cancel search command

### Alternate name

CancelSearch

### Function

It cancels the preview search operation.

### Availability

Execute this command after AGTFindFirstRecord to cancel the search operation

### Format

AGTCancelSearch C <Client> <ProcID> <InvokeID> § 0 ‡

### Data Parameters:

None

### Return Values

Code	Description
E28885	Not attached to a job. Attach a job.
E28907	Attached job is not a Managed Dialing job. Command is only available for Managed Dialing jobs.
E28908	Agent is not previewing a customer record. Retry after receiving a record to preview.
M00000	Complete.
E28909	Managed call already complete
E29954	Search is not in progress. To initiate the search, send command AGTFindFirstRecord.
S28833	Request is pending

## Examples

### Agent Application to Agent Binary

```
AGTCancelSearch      COrigID      PrID  InID 0 ‡
```

### Agent Binary to Agent Application

```
AGTCancelSearch  PAgent server 10450 111 2 §  
0 §  
S28833‡  
AGTCancelSearch  RAgent server 10450 111 2 §  
0 §  
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

‡ = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## AGTClearDataSet

AGTClearDataSet is a job setup command.

### Alternate Name

ClearDataSet

### Function

Clears all calling list fields included with call notification events. (Use AGTSetDataField to set fields.)

Use this command to change data fields during a job.

Blend jobs use both inbound and outbound calling lists. To clear the data field settings for a blend job, execute AGTClearDataSet twice: once for the inbound calling list and once for the outbound calling list.

After clearing the fields, specify a new set of calling list fields with AGTSetDataField.

It is not necessary to clear the data set immediately after attaching to a job. AGTDetachJob automatically clears the data fields set by AGTSetDataField.

### Availability

Execute this command any time after AGTAttachJob and after at least one execution of AGTSetDataField.

**Format**

AGTClearDataSet C <Client> <ProcID> <InvokeID> 1\$

<ListType> ‡

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameter**

ListType	Type of calling list used by the attached job: inbound (I) or outbound (O). Use upper case letters to identify the calling list type. Blend jobs use both I and O calling lists. Inbound jobs use type I calling lists. Outbound, Managed Dialing, Unit Work List, and Sales Verification jobs use type O calling lists.
----------	---

**Return values**

Code	Description
E28885	Not attached to a job. Execute AGTAttachJob and retry.
E28891	Must specify inbound or outbound operation. Retry the command with the <ListType> parameter set to either I or O.
E28892	There are no inbound calling list fields available. Retry with <ListType> O.
E28893	No outbound calling list fields available. Retry with <ListType> I.
M00000	Complete.

**Examples****Agent Application to Agent Binary**

```
AGTClearDataSet C COriginator_ID 11111 42 1$
O‡
```

**Agent Binary to Agent Application**

```
AGTClearDataSet R Agent server 2570 42 2$
O $
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**See also**

[AGTAttachJob](#)

[AGTDetachJob](#)

[AGTSetDataField](#)

## AGTConnHeadset

AGTConnHeadset is a connection command.

**Alternate Name**

ConnHeadset

**Function**

Connects a headset to Proactive Contact using direct-connect, dial-in, or dial-back headset connections. Once this command executes, Proactive Contact places a call to the extension identified by the reserved headset ID. This enables the voice connection to handle calls.

To reverse the effects of this command, execute AGTDisconnHeadset.

**Availability**

Execute this command after executing AGTReserveHeadset but before AGTAvailWork.

**Format**

AGTConnHeadset C <Client> <ProcID> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None.

**Return values**

Code	Description



Code	Description
E28850	Internal Proactive Contact system error: cannot open a channel to the operator monitor process; unable to connect headset ID. Indicates an Proactive Contact system problem.
E28872	Headset is already connected.
E28873	Headset ID is not reserved. Execute AGTReserveHeadset and retry.
E28874	A connect headset request is already pending.
E28875	The headset connect request did not register. Indicates an Proactive Contact system problem.
E28876	The headset is not connected. This message indicates that there is a problem with the telephone connection to the headset, or there is an Proactive Contact system problem.
E28920	§ <HeadsetID>† Headset ID is not in the reserved list. Retry the command with a valid reserved headset ID.
M00000	Complete.
S28833	Pending.

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Examples

#### Agent Application to Agent Binary

```
AGTConnHeadset C COriginator_ID 11111 87 0‡
```

#### Agent Binary to Agent Application

```
AGTConnHeadset P Agent server 8497 87 2§
0 §
S28833‡
AGTConnHeadset R Agent server 8497 87 2§
0 §
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**See also**

[AGTDisconnHeadset](#)

[AGTReserveHeadset](#)

## AGTDetachJob

AGTDetachJob is a system command.

**Alternate Name**

DetachJob

**Function**

Detaches the agent application from an active job on Proactive Contact. AGTDetachJob removes the availability of job-related commands. It also clears settings made with other job setup commands such as AGTSetUnit, AGTSetDataField, and AGTSetNotifyKeyField.

**Availability**

The agent application must have an attached job when AGTDetachJob executes. If the agent is available for work on a job (able to handle calls), then the application must execute AGTNoFurtherWork before issuing this command.

**Format**

AGTDetachJob C <Client> <ProcID> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None

**Return values**

Code	Description
E28913	No job is attached. There is nothing to detach.

Code	Description
E28914	Agent is still available for work on the job and calls are still being routed to the agent headset. Execute AGTNoFurtherWork and retry.
E28915	Not logged off the job yet. The AGTNoFurtherWork command is in the process of executing. Wait for a completion message from Proactive Contact and retry.
M00000	Complete.

## Examples

### Agent Application to Agent Binary

```
AGTDetachJob C COriginator_ID 11111 98 0‡
```

### Agent Binary to Agent Application

```
AGTDetachJob R Agent server 8497 98 2$
0 $
M00000‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty).

### See also

[AGTAttachJob](#)

[AGTNoFurtherWork](#)

## AGTDialDigit

AGTDialDigit is a working command.

### Alternate name

DialDigit

### Function

Causes Proactive Contact to send a DTMF tone representing a digit on an open telephone line.

Use this command to call customer telephone number extensions or numbers that require the agent to pause or send the \* or # tones.

(To hang up a call with a customer and keep the line open, execute AGTHangupCall or AGTManualCall.)

**Availability**

The command is available when the agent is working with a customer record and has an open telephone line. (An agent gets an open telephone line by receiving a call from a customer.)

This command is not available on an Avaya Proactive Contact with CTI system.

**Format**

AGTDialDigit C <Client> <ProclD> <InvokeID> 1§

<Digit> ‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty).

**Data parameter**

Digit	The digit for Proactive Contact to call. Use one character: a single number (0 - 9), *, or #. If the Proactive Contact master.cfg file contains values for the DIAL_STAR and DIAL_POUND variables, Proactive Contact substitutes those values for the normal * or # tones.
-------	--

**Return values**

Code	Description
E28866	A telephone line is not available. An open telephone line is only available after the agent receives a call.
E28867	The telephone line is not off-hook. Place the telephone line in off-hook state and retry.
E29950	This feature is not available on an Proactive Contact with CTI system.
M00000	Complete.

Code	Description
S28814	The transfer for the last call is in progress. Retry after the transfer is complete.

## Examples

### Agent Application to Agent Binary

```
AGTDialDigit C COriginator_ID 11111 235 1$
3‡
```

### Agent Binary to Agent Application

```
AGTDialDigit R Agent server 2570 235 2$
0 $
M00000‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTHangupCall](#)

[AGTManualCall](#)

[AGTReleaseLine](#)

## AGTDialerMode

AGTDialerMode is a working command. It returns the "SWITCHTYPE" parameter from the master.cfg file (the two valid values are "SOFTDIALER" and "DIGITAL").

### Alternate name

DialerMode

### Function

Use this command to know the mode in which the dialer is running.

### Availability

This command is available after the AGTLogon command executes.

### Format

AGTDialerMode C <Client> <ProcID> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### Data Parameters

None

#### Return values

Code	Description
SOFTDIALER	CTI dialer
DIGITAL	Non-CTI dialer
M00000	Complete.

#### Example

##### Agent Application to Agent Binary

AGTDialerMode C COriginator\_ID 11111 75 0‡

##### Agent Binary to Agent Application

AGTDialerMode R Agent server 2570 75 2§

0 §

M00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## AGTDisconnHeadset

AGTDisconnHeadset is a connection command.

#### Alternate name

DisconnHeadset

#### Function

Disconnects a headset.

Use this command as part of the agent logout process.

This command takes the headset ID from the Proactive Contact reserved headset list. See [AGTFreeHeadset](#).

**Availability**

Execute this command after receiving a completion message from the AGTNoFurtherWork command and before the AGTLogoff command. Executing AGTLogoff before this command forces a disconnect. If the agent application receives a call while the AGTNoFurtherWork command is pending and this command executes, the command disrupts agent headset service.

If this command executes when the agent application has no headset connection, the agent binary returns an error.

If this command executes when the agent application is available for work (see AGTAvailWork), work-related commands return an error until the agent application reconnects the headset.

**Format**

AGTDisconnHeadset C <Client> <ProclD> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None.

**Return values**

Code	Description
E28873	The headset is not reserved. There is no headset to disconnect.
E28876	The headset is not connected. There is no headset connection to close.
E28877	There is a disconnect headset request already pending.
E28900	§<Message>† An Proactive Contact system internal error occurred. Error message text follows.
M00000	Complete.

Code	Description
S28833	Pending.

## Examples

### Agent Application to Agent Binary

```
AGTDisconnHeadset C COriginator_ID 11111 65 0‡
```

### Agent Binary to Agent Application

```
AGTDisconnHeadset P Agent server 17916 65 2$
0 $
S28833‡
AGTDisconnHeadset R Agent server 17916 65 2$
0 $
M00000‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## See also

[AGTAvailWork](#)

[AGTConnHeadset](#)

[AGTNoFurtherWork](#)

[AGTReserveHeadset](#)

## AGTDoNotCall

AGTDoNotCall is a working command.

### Alternate Name

DoNotCall

### Function

Identifies a customer record as “do not call” (DNC). All matching records on other calling lists are also marked DNC.

The DNC feature ensures that records appearing in multiple calling lists will not be recalled.



### Availability

AGTDoNotCall is available when an agent is working with a customer record.

### Format

AGTDoNotCall

### Data parameters

None.

### Return values

None.

## AGTDumpData

AGTDumpData is a system command.

### Alternate name

DumpData

### Function

Dumps the memory structures for the agent application into an Proactive Contact file named <AgentName>\_<FileName>.dmp. The <AgentName> is the agent application user name. The agent application must submit a <FileName> with the command.

Use AGTDumpData after confirming its use with your Proactive Contact representative. Use this command for troubleshooting during application development. It allows you to check memory flags, status flags, and variable settings.

### Availability

This command is available any time after AGTLogon executes.

### Format

AGTDumpData C <Client> <ProclD> <InvokeID> 1§  
<FileName>‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameter**

FileName	The file name for the dump file. To use AGTDumpData repeatedly in a single customer session without overwriting the dump file, vary the file name. Use 7-bit USASCII, no embedded spaces or special characters. The <FileName> parameter is case sensitive.
----------	---

**Return value**

Code	Description
M00000	Complete.

**Output file sample**

Settings dumped to the output file depend on the system settings at the time the command executes.

Following is an example of AGTDumpData output.

```
Operator Info Structures: 01 #1
oper_jobname = 'mangjg'
oper_jobnum = '132'
oper_name = '01'
oper_pid = '852'
oper_ipcaddr = '1852'
oper_ipc_qtype = 'Q'
oper_log_type = 'M'
oper_assign_type = 'O'
oper_curr_type = ''
oper_unit_id = 'Allid'
oper_is_monitor = 'N'
oper_online = 'Y'
oper_ready = 'N'
oper_on_call = 'Y'
oper_call_time = '827524979'
oper_out_of_service= 'N'
oper_long_talk1 = 'N'
oper_long_talk2 = 'N'
oper_release_line = 'Y'
oper_release_time = '827525889'
oper_marked_on_job = 'Y'
oper_marked_off_job= 'N'
oper_logout_req = 'N'
oper_logout_permitted= 'N'
oper_start_clk = '827524041'
oper_last_tran = 'O'
oper_last_idle = 'O'
oper_idle_time = '827525889'
oper_mesg_cnt = '0'
oper_rec_pos = '-1'
```

## Avaya Proactive Contact Agent API 5.2

```
oper_mesg = ''
oper_info_mesg = ''
oper_tel_line = '-1'
oper_mon_line = '-1'
oper_headset_id = ''
oper_device_name = '/dev/pty/ttyv1'
oper_transferjob_req = 'N'
oper_transferjob = ''
oper_curr_stat = 'I'
oper_syslogintime = '827523973'
oper_releasedtime = '0'
oper_acquiredtime = '0'
oper_offlinetime = '827523973'
Extended Operator Statistics:
bac_callsworked = 0
bac_idlecount = 0
bac_idletime = 0
inb_callsinwait = 0
inb_callsworked = 0
inb_idlecount = 0
inb_idletime = 0
inb_talktime = 0
inb_updatetime = 0
inb_worktime = 0
job_callsworked = 0
job_idlecount = 0
job_idletime = 0
job_talktime = 0
job_updatetime = 0
job_worktime = 0
out_callsworked = 0
out_idlecount = 0
out_idletime = 0
out_talktime = 0
out_updatetime = 0
out_worktime = 0
previewtime = 0
operstats_ptr->out_codes = 0
operstats_ptr->inb_codes = 0
Operator Headset status(if one is reserved) -1:
Operator PCSTATE :
client_connected = 1
client_signed_on = 1
datashm_attached = 1
unitshm_attached = 1
dictshm_attached = 1
headset_reserved = 1
headset_active = 1
headset_hungup = 0
base_attached = 1
conn_mgr_ipc = 1
caller_ipc = 1
porter_ipc = 0
input_ipc = 1
in_abort = 0
iichb_login = 0
Operator AGTSTATE :
attached_job = 1
```

## Avaya Proactive Contact Agent API 5.2

```
avail_for_work = 1
ready_next_work_item = 0
on_work_item = 0
telephone_line_avail = 0
telephone_offhook = 0
no_further_work_pending = 0
no_further_work_set = 0
Operator AGENT_CB :
caller_ipc = 6
porter_ipc = 12
curr_assign = 0
curr_call_type = 0
curr_phone = 1
finalstat =
idle_type = 0
op_mesg = JOHN DOE (Preview)
op_name = 01
op_unit_id = Allid
recall_name =
recall_date =
recall_phone =
recall_time =
listbuf = listgjjg
ilistbuf =
out_notify_field = 3
inb_notify_field = -1
outlist_cnt = 40
inblist_cnt = -1
curr_jobnum = 132
job_infonum = 1
maxrecsize = 2048
phone_total = 2
portnum = -1
testmodeflg = 1
idle_time = 528
busytime = 910
contime = 22
idle_clock = 827525889
log_time = 0
preview_time = 30
talking_time = 0
port_base = 1750
pqtype = S
jobname = mangjjg
oper_calltype =
headset_id = 1
headset_id_pending =
device_name = /dev/pty/ttyv1
acct_ownership = 0
inboundjob = 0
inboundonly = 0
pv_dial = 1
pv_length = -1
pv_cancel = 0
rec_num = 0
connectf = 1
acd_agent = 0
op_num = 1
```

## Avaya Proactive Contact Agent API 5.2

```
op_hs_slot_num = -1
op_hid = 1
op_equip = 240
ireclen = 298
reclen = 298
transfer_in_prog = 0
cntry_code_loc = -1
icntry_code_loc = -1
tz_loc[0] = 0
tz_loc[1] = 249
tz_loc[2] = 251
tz_loc[3] = 0
tz_loc[4] = 0
tz_loc[5] = 0
tz_loc[6] = 0
tz_loc[7] = 0
tz_loc[8] = 0
tz_loc[9] = 0
tz_loc[10] = 0
tz_loc[11] = 0
tz_loc[12] = 0
tz_loc[13] = 0
tz_loc[14] = 0
```

## Examples

### Agent Application to Agent Binary

```
AGTDumpData C COriginator_ID 11111 71 1$
dmp0305‡
```

### Agent Binary to Agent Application

```
AGTDumpData R Agent server 5846 71 2$
0 $
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTLogloStart](#)

## AGTEchoOff

AGTEchoOff is a system command used with the **-t** option only.

### Alternate Name

EchoOff

## Function

Turns off echoing of characters to the screen.

Use this command to verify commands and data parameters for the agent application. The default state when the agent binary starts is **echo on**. During an agent command line interface session, this command turns off echoing of characters to the screen. (You might wish to write a developer GUI that can display echoed commands in a window.)

## Availability

This command is available any time after the AGTLogon command executes.

## Format

AGTEchoOff C <Client> <ProclD> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

## Data parameters

None.

## Return value

Code	Description
M00000	Complete.

## Examples

### Agent Application to Agent Binary

```
AGTEchoOff C COriginator_ID 11111 32 0‡
```

### Agent Binary to Agent Application

```
AGTEchoOff R Agent server 5846 32 2§  
0 §  
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**See also**[AGTEchoOn](#)

## AGTEchoOn

AGTEchoOn is a system command used with the **-t** option only.

**Alternate name**

EchoOn

**Function**

Turns on echoing of characters to the screen. Use this command to verify commands and data parameters for the agent application. You can write a developer GUI that can display echoed commands in a window. The default state when the agent binary starts is echo on.

**Availability**

This command is available any time after the AGTLogon command executes.

**Format**

AGTEchoOn C <Client> <ProclD> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Return value**

Code	Description
M00000	Complete.

**Examples****Agent Application to Agent Binary**

```
AGTEchoOn C COriginator_ID 11111 162 0‡
```

**Agent Binary to Agent Application**

```
AGTEchoOn R Agent server 5846 162 2§  
0 §
```

M00000‡

**See also**

[AGTEchoOff](#)

## AGTFindFirstRecord

AGTFindFirstRecord is a working command.

**Alternate name**

FindFirstRecord

**Function**

This method will be used to initiate search in the preview. In response to this call the agent can get the number of records found, index of the searched record and its data.

**Availability**

Execute this command to initiate the preview search after the preview notification.

**Format**

AGTFindFirstRecord C <Client> <ProcID> <InvokeID> 1 § <KeyVal> ‡

**Data parameters**

KeyVal	This the key value for the key that is set in PVKEYFLD in job file. If we have the Key as CITY then the sKeyVal will be a city name.
--------	--

**Return values**

Code	Description
E28885	Not attached to a job. Attach a job.
E28907	Attached job is not a Managed Dialing job. Command is only available for Managed Dialing jobs.
E28908	Agent is not previewing a customer record. Retry after receiving a record to preview.
E28909	Managed call already complete



Code	Description
M00000	Complete.
M00001	Data record
E29953	PVRS (Preview search) is not enabled. Configure the preview search properly.
E12156	Can not find the record with the search key val.
E29955	Search value (keyval) is not entered.
E29956	Search already is in progress. The command AGTFindFirstRecord already sent and the search is initiated. Send AGTFindNextRecord to search next or previous record.
S28833	Request is pending

## Examples

### Agent Application to Agent Binary

```
AGTFindFirstRecord COrigID          PrID  InID 1  $
JOHN DOE#
```

### Agent Binary to Agent Application

```
AGTFindFirstRecord PAgent server    10450 111 2  $
0 $
S28833#
AGTFindFirstRecord DAgent server    10450 111 3  $
0 $
M00001 $
S12153,501,1#
AGTFindFirstRecord DAgent server    10450 111 5  $
0 $
M00001 $
501 records found, 1 is being displayed $
MANAGED $
ACCTNUM,4302209780005280#
AGTFindFirstRecord DAgent server    21740 111 3  $
0 $
M00001 $
PHONE1,4255521009#
AGTFindFirstRecord RAgent server    10450 111 2  $
```

0 §  
M00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## AGTFindNextRecord

AGTFindNextRecord is a working command.

### Alternate name

FindNextRecord

### Function

It is used to get next or previous record after the search is initiated.

### Availability

Execute this command after the AGTFindFirstRecord to find previous or next record.

### Format

AGTFindNextRecord C <Client> <ProcID> <InvokeID> 1 § <sNext> ‡

### Data parameters

sNext	If the value of sNext is P then it will display the previous record otherwise it will display the next record.
-------	--

### Return values

Code	Description
E28885	Not attached to a job. Attach a job.
E28907	Attached job is not a Managed Dialing job. Command is only available for Managed Dialing jobs.
E28908	Agent is not previewing a customer record. Retry after receiving a record to preview.

Code	Description
E28909	Managed call already complete
M00000	Complete.
M00001	Data record
E29953	PVRS (Preview search) is not enabled. Configure the preview search properly.
E12152	Previous record is not present in the searched records.
E12153	No more record found. Next record is not present in the searched records.
E29954	Search is not in progress. To initiate the search, send command AGTFindFirstRecord and then try with this command.
S28833	Request is pending

## Examples

### Agent Application to Agent Binary

```
AGTFindNextRecord COrigID          PrID  InID 1  $
N#
```

### Agent Binary to Agent Application

```
AGTFindNextRecord PAgent server      10450 111 2  $
0 $
S28833#
AGTFindNextRecord DAgent server      10450 111 3  $
0 $
M00001 $
S12153,501,2#
AGTFindNextRecord DAgent server      10450 111 5  $
0 $
M00001 $
501 records found, 2 is being displayed $
MANAGED $
ACCTNUM,4302209860005663#
AGTFindNextRecord DAgent server      21740 111 3  $
0 $
M00001 $
```

## Avaya Proactive Contact Agent API 5.2

```
PHONE1,4255521009#  
AGTFindNextRecord  RAgent server      10450 111 2    $  
0 $  
M00000#
```

### Agent Application to Agent Binary

```
AGTFindNextRecord  COrigID              PrID  InID 1    $  
P#  
Agent Binary to Agent Application  
AGTFindNextRecord  PAgent server      10450 111 2    $  
0 $  
S28833#  
AGTFindNextRecord  DAgent server      10450 111 3    $  
0 $  
M00001 $  
S12153,501,1#  
AGTFindNextRecord  DAgent server      10450 111 5    $  
0 $  
M00001 $  
501 records found, 1 is being displayed $  
MANAGED $  
ACCTNUM,4302209780005280#  
AGTFindNextRecord  DAgent server      21740 111 3    $  
0 $  
M00001 $  
PHONE1,4255521009#  
AGTFindNextRecord  RAgent server      10450 111 2    $  
0 $  
M00000#
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## AGTFinishedItem

AGTFinishedItem is a working command.

### Alternate name

FinishedItem

### Function

Use this command to release a customer record and insert a call completion code into the record. If the agent is on the telephone with the customer, this command also executes AGTReleaseLine to end the call. If Managed Dialing preview call cancellation is enabled, use this command to close the customer record without placing a call.

After AGTFinishedItem executes, the agent does not receive another call or preview message until AGTReadyNextItem executes. The agent application can use the time between the two commands to change agent or job settings before taking another call or preview message. If the agent sent an AGTNoFurtherWork request while working on the customer record, AGTFinishedItem sends the request to Proactive Contact after releasing the record.

In case of Agent blending (PAB), AGTlicbOffline message can come right after AGTFinishedItem if any inbound calls are in the queue and the agent is identified by the blend engine to handle the inbound call.

**Availability**

This command is available only when the agent is working with a customer record.

**Format**

```
AGTFinishedItem C <Client> <ProclD> <InvokeID> 1§  
<CompCode>‡
```

**Data parameter**

CompCode	The three-digit numeric code to place in the customer record that indicates the results of the telephone call. The code must be from the list returned by the AGTListKeys command. Completion code 98 identifies the call as Agent Owned Recall.
----------	--

**Return values**

Code	Description
E28866	There is no line available. This error generally indicates the agent is trying to cancel a Managed Dialing call too late.
E28885	The agent application is not attached to a job. Execute AGTAttachJob and retry.
E28919	The agent is not currently working with a customer record. There is no customer record to release.
E28947	Completion code is invalid. Retry command with a code returned by AGTListKeys for the current job.
M00000	Complete.

Code	Description
S28833	Pending.

## Examples

### Agent Application to Agent Binary

```
AGTFinishedItem C COriginator_ID 11111 24 1$
98‡
```

### Agent Binary to Agent Application

```
AGTFinishedItem R Agent server 28705 24 2$
0 $
M00000‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTCallNotify](#)

[AGTListKeys](#)

[AGTReadyNextItem](#)

[AGTReleaseLine](#)

## AGTFreeHeadset

AGTFreeHeadset is a connection command.

### Alternate name

FreeHeadset

### Function

Frees the reserved headset ID by:

Removing the ID from the appropriate tables on Proactive Contact

Breaking the link between the agent application workstation and the telephone extension associated with the headset ID

AGTFreeHeadset removes the association between the workstation and the telephone extension by flushing the memory buffer for the headset ID.

### Availability

Execute this command after the AGTDisconnHeadset command and before the AGTLogoff command.

### Format

AGTFreeHeadset C <Client> <ProclD> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

None.

### Return values

Code	Description
E28873	There is no headset reserved. There is nothing to free.
E28879	The headset is not disconnected. Disconnect the headset and retry.
M00000	Complete.
S28833	Pending.

### Examples

#### Agent Application to Agent Binary

```
AGTFreeHeadset C COriginator_ID 11111 561 0‡
```

#### Agent Binary to Agent Application

```
AGTFreeHeadset R Agent server 17916 561 2§  
0 §  
M00000‡
```

### See also

[AGTDisconnHeadset](#)

[AGTReserveHeadset](#)

## AGTGetHeadsetVol

AGTGetHeadsetVol is a working command.

### Alternate name

GetHeadsetVol

### Function

Use this command to get the current headset volume settings before changing them with AGTAdjustHeadset. The first data value returned is the ear volume, the second is the mouth volume.

### Availability

This command is available when the agent is working with a customer record and has an open telephone line. Releasing the telephone line disables this command. An agent gets an open telephone line by receiving a call from a customer. This setting is only available for direct connect headsets on Proactive Contact systems using OLIC cards.

### Format

AGTGetHeadsetVol C <Client> <ProcID> <InvokeID> 0 ‡

### Data parameters

None.

### Return values

Code	Description
E28850	Cannot open a channel to the operator monitor process. Indicates an Proactive Contact system problem.
E28851	There is no response from the operator monitor process. Indicates an Avaya system problem.
E28866	A telephone line not available. Retry with an open telephone line.
E28867	The open telephone line is not off-hook. Retry with an off-hook telephone line.
M00000	Complete.



Code	Description
M00001	§ <Ear> § <Mouth>‡ Data message. The current volume settings for the headset earphone and microphone. Range for settings is from one to eight.
S28814	A transfer is in progress. The customer call is still in the process of being transferred to the agent. Retry after connecting to the call.

## Examples

### Agent Application to Agent Binary

```
AGTGetHeadsetVol C COriginator_ID 11111 64 0‡
```

### Agent Binary to Agent Application

```
AGTGetHeadsetVol D Agent server 29722 64 4 §
0 §
M00001 §
1 §
4‡
AGTGetHeadsetVol R Agent server 29722 64 2§
0 §
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

‡ = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## See also

[AGTAdjustHeadset](#)

[AGTHangupCall](#)

[AGTManualCall](#)

## AGTGetScreen

AGTGetScreen is a working command.

## Alternate name

GetScreen

## Function

The dialer side of the Agent Screens application encompasses two distinct parts. The first is the creation of screen definitions for all screens associated with a job when an agent joins the job. These definitions are stored in memory (character arrays) while the agent remains attached to the job. Requests for information about a particular screen are answered by returning the array that defines the screen. This way, each screen is processed only once when the agent joins the job instead of each time an agent requests it.

Screen definitions are composed of two element types, fields and labels. Fields are the sections of screen real estate where customer information is displayed or entered. Labels are used to name the fields or provide other static information to the agent. Labels do not change as different customer records are displayed on the screen.

A field record is comprised of five segments: element type (F), x position, y position, width, and text. Each segment is described below:

- | Type. F for fields, L for labels.
- | X position. Place on a 24 column screen where the field starts. Column 1 is on the left.
- | Y position. Place on an 80 row screen where the field starts. Row 1 is on the top.
- | Width. The width in number of characters.
- | Text. The text segment contains the field name and 6 attributes separated by colons. The entire segment is enclosed in double quotes.

1. Name. The field name.

2. Cursor Access. 1 = cursor might move to the field. 0 = cursor might not move to the field.

3. Locked. 1 = locks the field. 0 = lets agents enter data.

4. Data format. C = Alpha-numeric. N = Numeric. D = Date. T = Time. \$= Decimal.

5. Acceptable entries. Comma separated list of acceptable values.

6. Entry required. 1 = data must be entered. 0 = input not required.

7. Dot suppression. 0 = dots displayed as field filler. 1 = no dots.

The field record -- F, 9, 12, 4, "CARDTYPE:1:0:C:Visa, MC, Amex:0:1" -- defines a four character alpha-numeric field named CARDTYPE at column 9, row 12. The cursor can move to the field and the agent can enter "Visa", "MC", or "Amex". Input is not required and blank or partial fields are not filled with dots.

A label record is also comprised of five segments: element type (L), x-position, y-position, label length, and label. Position values follow the same rules as for fields. The label length is the number of characters in the label name.

Label record -- L, 4, 12, 5, "Card:" -- is a suitable label for the field defined above. It starts in the fourth column of the same row occupies five characters before the field.

Note: there is a one character space between the label and the field.

There is a record separator (character 30) between each of the records. All of the field records will come first, followed the label records.

The second part of the Agent Screens application involves the Agent API commands used to screen lists and individual screen definitions. Two new commands are introduced for these tasks.

AGTListScreens( I | O) returns the list of inbound or outbound screens for the current job. Each screen name is followed by a record separator.

AGTGetScreen(name) returns the screen definition data described above for the specified screen.

### Example

```
ACLIENT --> AGTGetScreen          COrigID          MoAgt 902      1^^list1^C
SERVER <-- AGTGetScreen          DAgent server      3455  902 39
^^0^^M00001^^list1^^F, 9,16, 0
,"NAME1:1:0:C::0:1"^^F,
9,17,30,"NAME2:1:0:C::0:1"^^F,52,16,18,"ACCTNUM:1:0:C::0:1"^^F,47,17,
0,"BALANCE:0:1:C::0:1"^^F,47
,18, 0,"DELQUENT:0:1:C::0:1"^^F,54,22, 0,"BEHSCORE:0:1:C::0:1"^^F,70,18,
0,"DAYS:0:1:C::0:1"^^F, 9,18,12,"PHONE1:1:0:C::0:
1"^^F, 9,19,12,"PHONE2:1:0:C::0:1"^^F,70,17, 0,"CREDLINE:0:1:C::0:1"^^F,70,19,
0,"PAYDAY:0:1:C::0:1"^^F,47,19, 0,"PAYAMT:0
:1:C::0:1"^^F, 9,20, 0,"ZIPCODE:0:1:C::0:0"^^F,47,20,
0,"INTERNAL:0:1:C::0:1"^^F,70,20, 0,"EXTERNAL:0:1:C:A,a:0:1"^^L, 1,1
6, 6,"Name1:"^^L,36,16, 7,"Account"^^L,44,16, 7,"Number:"^^L, 1,17,
6,"Name2:"^^L,36,17, 8,"Balance:"^^L,59,17, 7,"Credit:
"^^L, 1,18, 5,"Home:"^^L,36,18, 3,"Del"^^L,40,18, 4,"Amt:"^^L,59,18,
4,"Days"^^L,64,18, 4,"Del:"^^L, 1,19, 5,"Work:"^^L,36
,19, 4,"Last"^^L,41,19, 4,"Amt:"^^L,59,19, 4,"Last"^^L,64,19, 4,"Pmt:"^^L,
1,20, 4,"Zip:"^^L,36,20, 9,"Internal:"^^L,59,20
, 9,"External:"^^L,36,22, 8,"Behavior"^^L,45,22, 6,"Score:"^C
SERVER <-- AGTGetScreen          RAgent server      3455  902 2      ^^0^^M00000^C
```

## AGTHangupCall

AGTHangupCall is a working command.

### Alternate name

HangupCall

### Function

Use this command to hang up a customer call before using a command that requires an open telephone line (such as AGTGetHeadsetVol or AGTManualCall). The AGTManualCall command executes AGTHangupCall (if necessary) before calling the number for the manual call.

### Availability

This command is available when the agent is working with a customer record and is talking with the customer.

### Format

AGTHangupCall C <Client> <ProcID> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

None.

### Return values

Code	Description
E28866	A telephone line is not available. There is no call to hang up.
M00000	Complete.

### Examples

#### Agent Application to Agent Binary

```
AGTHangupCall C COriginator_ID 11111 75 0‡
```

#### Agent Binary to Agent Application

```
AGTHangupCall R Agent server 2570 75 2§  
0 §  
M00000‡
```

### See also

[AGTDialDigit](#)

[AGTManualCall](#)

[AGTReleaseLine](#)

## AGTHeadsetConnBroken

AGTHeadsetConnBroken is a notification event message from the server.

### Function

Notifies the agent application that no agent headset connection exists. The headset might have become physically disconnected or the agent might have hung up the telephone line manually. When a headset disconnects, the agent binary changes the

agent state to “headset broken” and blocks the agent application from any commands that permit the agent to accept another call. Before the agent can resume handling calls, restore the physical headset connection (if necessary) and re-establish the headset connection with AGTConnHeadset. The agent must follow the steps for Log out of Proactive Contact and then Log in to Proactive Contact to re-establish the headset connection with AGTConnHeadset. (Logging out/Logging in of Proactive Contact steps are mentioned in the Agent Tasks Overview section).

---

This message does not post when the agent application executes the AGTDisconnHeadset command.

### Format

AGTHeadsetConnBroken N <Agentserver> <ProcID> <InvokeID>§ 2 §

1 §E28800‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

None.

### Return values

Code	Description
E28880	Headset connection is broken.
E28881	Headset is reconnected.
E28900	§ <message>‡ An Proactive Contact system internal error occurred. Error message text follows.

### Examples

```
AGTHeadsetConnBroken N Agent server 22486 0 2§
1 §
E28880‡
```

### See also

[AGTConnHeadset](#)

[AGTDisconnHeadset](#)

## AGTHoldCall

AGTHoldCall is a working command.

### Alternate name

HoldCall

### Function

Use this command to place a customer call on hold while an agent performs other activities. Releasing the telephone line with either AGTReleaseLine or AGTFinishedItem disables this command.

### Availability

This command is available when the agent is working with a customer record and is talking with the customer.

This command is not available on an Avaya Proactive Contact with CTI system.

### Format

AGTHoldCall C <Client> <ProcID> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

None

### Return values

Code	Description
E28866	Telephone line is not available. There is no open telephone line.
E28867	The open telephone line is not off-hook. Retry with an off-hook telephone line. This message means that the agent has already placed the call on hold.
E29950	This feature is not available on an Proactive Contact with CTI system.
M00000	Complete.

Code	Description
S28814	Transfer is in progress. Proactive Contact cannot place the call on hold.

## Examples

### Agent Application to Agent Binary

```
AGTHoldCall C COriginator_ID 11111 98 0‡
```

### Agent Binary to Agent Application

```
AGTHoldCall R Agent server 2570 98 2$
0 $
M00000‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTFinishedItem](#)

[AGTReleaseLine](#)

[AGTUnholdCall](#)

## AGTHookflashLine

AGTHookflashLine is a working command.

### Alternate Name

HookflashLine

### Function

Sends a hookflash and places a manual call.

Use it to transfer calls on Proactive Contact systems connected to a switch that accepts hookflash-initiated transfers. In these systems, the hookflash places the customer on hold.

After the transfer is complete, the agent application must execute one of the following commands. The agent can speak to the person receiving the transferred call before executing any of the follow-up commands.

AGTHookflashLine a second time with no <PhoneNumber> parameter to establish a conference call

AGTReleaseLine to disconnect the agent's telephone connection but maintain contact with the customer record

AGTFinishedItem to disconnect the agent's telephone connection and release the customer record

### Availability

This command is available when the agent is working with a customer record and is talking with the customer.

This command is not available on an Avaya Proactive Contact with CTI system.

### Format

AGTHookflashLine C <Client> <ProclD> <InvokeID> 1§

<PhoneNumber> ‡

AGTHookflashLine C <Client> <ProclD> <InvokeID> 1§

‡

### Data Parameter

PhoneNumber	The phone number to call. Numeric, up to 43 characters, no embedded spaces or special characters. If the number is a complete telephone number rather than an extension, the format must match the standard format in the phonefmt.cfg configuration file on the Proactive Contact system.
-------------	--

### Return Values

Code	Description
E28866	Telephone line is not available. Proactive Contact cannot place the call. Retry later.
E28867	Telephone line not off-hook. Retry with an off-hook open line.
E29950	This feature is not available on an Proactive Contact with CTI system.
S28814	Transfer is in progress. Retry after the transfer is complete.



## Examples

### Agent Application to Agent Binary

```
AGTHookflashLine C COriginator_ID 11111 164 1$  
3424‡
```

### Agent Binary to Agent Application

```
AGTHookflashLine R Agent server 2570 164 2$  
0 $  
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See Also

[AGTManualCall](#)

[AGTTransferCall](#)

## AGTlicbAbort

AGTlicbAbort is a notification event message from the server.

### Alternate name

licbAbort

### Function

This message only appears on Agent Blending systems for Proactive Contact agents logged in as ACD agents.

Proactive Contact sends this message to inform the agent application that a pending acquisition or release has terminated abnormally.

Completed acquisitions and releases result in AGTlicbOnline or AGTlicbOffline notifications.

### Availability

This notification event message appears after the agent application receives AGTlicbFeNotif.

### Format

```
AAGTlicbAbort N <Agent server> <ProcID> <InvokeID> 2$
```

```
0 § M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### Data parameters

None

#### Return values

M00000 Complete.

#### Examples

##### Agent Binary to Agent Application

```
AGTlicbAbort N Agent server 7924 17 2$  
  0 $  
M00000‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

#### See also

[AGTlicbFeNotif](#)

[AGTlicbOffline](#)

[AGTlicbOnline](#)

## AGTlicbFeNotif

AGTlicbFeNotif is a notification event message from the server.

#### Alternate name

licbFeNotif

#### Function

This message only appears on Agent Blending systems for Proactive Contact agents logged in as ACD agents.

Proactive Contact sends this message to inform the agent application that the ACD agent is being acquired for outbound calling or released to inbound calling.

It should display an appropriate message on the agent's screen.

#### Availability

The agent application receives this message immediately before AGTlicbOnline or immediately after AGTlicbOffline.

### Format

AAGTlicbFeNotif N <Agent server> <ProcID> <InvokeID 2§

0 § S28971‡

AGTlicbFeNotif N <Agent server> <ProcID> <InvokeID 2§

0 § S28972‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

None

### Return values

Code	Description
S28971	Standby to take outbound calls... Acquisition to outbound is pending.
S28972	Standby to take inbound calls... Release to inbound is pending.

### Examples

#### Agent Binary to Agent Application

```
AAGTlicbFeNotif N Agent server 4437 0 2§  
0 §  
S28972‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTlicbOffline](#)

[AGTlicbOnline](#)

## AGTlicbOffline

AGTlicbOffline is a notification event message from the server. The agent application must execute AGTReadyNextItem before routing any inbound calls to the agent.

If you are using Avaya Proactive Contact with Avaya CT mode, the following commands will return the following error codes:

- AGTHoldCall - E29950 - not avail in softdialer mode
- AGTUnholdCall - E29950 - not avail in softdialer mode
- AGTManualCall - E29950 - not avail in softdialer mode
- AGTHookflashLine - E29950 - not avail in softdialer mode
- AGTTransferCall - E29950 - not avail in softdialer mode
- AGTDialDigit - E29950 - not avail in softdialer mode
- AGTAdjustHeadset - E29950 - not avail in softdialer mode
- AGTReadyNextItem - E28964 if the agent phone is busy (i.e., has a stray call on it), also returns E28965 if the Avaya CT link is down.
- AGTManagedCall - E28964 if the agent phone is busy (i.e., has a stray call on it), also returns E28965 if the Avaya CT link is down.

### Alternate name

licbOffline

### Function

This message only appears on Agent Blending systems for Proactive Contact agents logged in as ACD agents.

For the agent release to inbound, Proactive Contact sends this message to inform the agent application that the agent release to the ACD for inbound calling is complete.

### Availability

This message arrives just before the AGTlicbFeNotif notifies the agent to prepare to receive inbound calls.

If the inbound call is answered in the same time frame by another agent, the agent may be returned to outbound with the AGTlicbOnLine message. No AGTlicbFeNotif occurs in this case.

### Format

```
AGTlicbOffline N <Agent server> <ProcID> <InvokeID> 2§  
0 § M00000†
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### Data parameters

None

#### Return values

Code	Description
M00000	Complete.

#### Examples

##### Agent Binary to Agent Application

```
AAGTlicbOffline N Agent server 9736 0 2$
0 $
M00000‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### See also

[AGTlicbFeNotif](#)

[AGTlicbOnline](#)

## AGTlicbOnline

AGTlicbOnline is a notification event message from the server.

#### Alternate name

licbOnline

#### Function

This message only appears on Agent Blending systems for Proactive Contact agents logged in as ACD agents.

Proactive Contact sends this message to inform the agent application that Proactive Contact acquired the agent for outbound calling.

### Availability

This message arrives after the AGTlicbFeNotif notifies the agent to prepare to receive outbound calls.

### Format

AGTlicbOnline N <Agent server> <ProclD> <InvokeID> 2\$  
0 § M00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

None

### Return values

Code	Description
M00000	Complete.

### Examples

#### Agent Binary to Agent Application

```
AAGTlicbOnline N Agent server 2478 0 2$  
0 §  
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTlicbFeNotif](#)

[AGTlicbOffline](#)

[AGTLogonAcd](#)

## AGTJobEnd

AGTJobEnd is a notification event message from the server.

### Alternate name

JobEnd

### Function

The Proactive Contact message sent to all agents currently logged in to a job when that job stops. This event message does not differentiate between jobs that end normally and jobs a supervisor stops manually.

If the agent is working, AGTJobEnd initiates AGTNoFurtherWork. The agent application must detach the job then log out or attach another job.

If job linking is enabled and the current job links to another job, the agent application receives an AGTJobTransLink notification instead of AGTJobEnd.

### Format

```
AGTJobEnd N <Agentserver> <ProclD> <InvokeID> 2$  
0 § M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

None

### Return values

Code	Description
M00000	Complete.

### Examples

#### Agent Binary to Agent Application

```
AAGTJobEnd N Agent server 17970 0 2$  
0 §  
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

[AGTAttachJob](#)

[AGTDetachJob](#)

[AGTJobTransLink](#)

AGTJobTransLink

AGTJobTransLink is a notification event message from the server.

Alternate name

JobTransLink

Function

This message only appears on Proactive Contact systems where job linking is enabled.

Use this message to acquire the job name to transmit with the AGTAttachJob command.

The message notifies the agent application that the current job is ending, and there is a job linked to it. The agent application should transfer the agent to the new job.

The agent binary sends this message to the agent application in place of the AGTJobEnd message.

If the agent is working with a customer record, AGTJobTransLink automatically initiates an AGTNoFurtherWork on behalf of the agent application.

The agent application must detach the job that is ending before attaching the linked job.

Format

AGTJobTransLink N <Agentserver> <ProcID> <InvokeID> 3§  
0 § M00000 § <JobName>‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

JobName	The value is the name of the job linked to the current job. Use up to 19 characters, 7-bit USASCII. The <JobName> is case sensitive and might not include special characters or embedded spaces.
---------	--



**Return values**

Code	Description
E28900	§ <message>‡ An Proactive Contact system internal error occurred. Error message text follows.
E28917	No job is attached. There is no current job that could have a job linked to it.
M00000	Complete.

**Examples****Agent Binary to Agent Application**

```

AGTJobTransLink N Agent server 17970 0 3§
0 §
M00000 §
outbndl‡

```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**See also**

[AGTAttachJob](#)

[AGTDetachJob](#)

[AGTJobEnd](#)

## AGTJobTransRequest

AGTJobTransRequest is a notification event message from the server.

**Alternate name**

JobTransRequest

**Function**

Notifies the agent application of a supervisor-initiated request to transfer the agent to another job.

Use this message to collect the job name to transmit with the AGTAttachJob command.

When a supervisor requests an agent transfer, the agent binary sends AGTJobTransRequest to the agent application after receiving the AGTReadyNextItem

command. The agent binary initiates an AGTNoFurtherWork command automatically on behalf of the agent application. Once the agent binary sends an AGTNoFurtherWork complete response to the agent application, the agent application can finish the job transfer by detaching the current job and attaching the new one.

Error messages can appear with this event name:

- If a supervisor initiates a job transfer request for an agent that is not attached to a job.
- If the agent has already executed AGTNoFurtherWork at the time of the transfer request.

### Format

```
AGTJobTransRequest N Agentserver <ProclD> <InvokeID> 3§
0 § M00000 § <JobName>‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

<JobName>	The value is the name of the job the agent should transfer to. Use up to 19 characters, 7-bit USASCII. The <JobName> is case sensitive and might not include special characters or embedded spaces.
-----------	---

### Return values

Code	Description
E28900	§ <Message> ‡ An Proactive Contact system internal error occurred. Error message text follows.
E28901	Agent is not available for work. The AGTNoFurtherWork command initiated by agent is not necessary.
E28917	No job is attached. The transfer request is inappropriate.
M00000	Complete.

## Examples

### Agent Binary to Agent Application

```
AGTJobTransRequest N Agent server 4880 0 3$
0 $
M00001 $
outgjjg‡
AGTJobTransRequest N Agent server 4880 0 2$
0 $
M00000 $
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTAttachJob](#)

[AGTDetachJob](#)

[AGTNoFurtherWork](#)

## AGTListCallbackFmt

AGTListCallbackFmt is a working command.

### Alternate name

ListCallbackFmt

### Function

Use this command to determine the date format used when sending date information to Proactive Contact and to determine the range of the phone number index for recalls.

### Availability

AGTListCallbackFmt is available any time a job, other than an inbound job, is attached.

Neither an agent nor Proactive Contact can recall a customer during an inbound job. Executing AGTListCallbackFmt on an inbound job results in an error.

### Format

```
AGTListCallbackFmt C <Client> <ProcID> <InvokeID> 0‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None

**Return values**

Code	Description
E28868	No recalls during inbound jobs. The current attached job is an inbound job. There is no recall format.
E28885	The agent application is not attached to a job. Execute AGTAttachJob and retry.
M00000	Complete.
M00001	§ <Format> § <Phones>àData message. <Format> is the date format used on Proactive Contact. * <Phones> is the number of phones in the customer record available for recall. This provides a range (from 1 to <Phones>) of index values to use with AGTSetCallback.

**Examples****Agent Application to Agent Binary**

```
AGTListCallbackFmt C COriginator_ID 11111 17 0‡
```

**Agent Binary to Agent Application**

```
AGTListCallbackFmt D Agent server 29722 17 4§
0 §
M00001 §
1999/03/06 §
2‡
AGTListCallbackFmt R Agent server 29722 17 2§
0 §
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**See also**

[AGTSetCallback](#)

## AGTListCallFields

AGTListCallFields is a system command.

**Alternate name**

## ListCallFields

## Function

Lists the data fields in a calling list.

Use this command to identify field names to use with the AGTSetDataField, AGTSetNotifyKeyField, or AGTReadField commands.

You can also use it to get field parameters to use with `AGTUpdateField`.

To list the fields for a particular job, use `AGTListDataFields` after attaching a job.

## Availability

AGTListCallFields might execute any time after AGTLogon executes.

### Format

```
AGTListCallFields C <Client> <ProclD> <InvokeID> 1$
                <ListName>±
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

± = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## Data parameters

ListName	Enter the name of the calling list. Use alphanumeric values, 7-bit USASCII. Calling list names are case sensitive.
----------	--

## Return values

Code	Description
E00518	<p>&lt;agent&gt; &lt;ListName&gt;</p> <p>There is no calling list with &lt;ListName&gt; on the system. The agent entry is the name of the process on Proactive Contact returning the error (it will always be agent). &lt;ListName&gt; is the name submitted by the client process.</p>

Code	Description
E70000	Incorrect number of arguments. Either the tenant name or the calling list is not passed.
E28644	<agent> <ListName> <TenantName> The input <ListName> does not belong to the Tenant that was set.
M00000	Complete.
M00001	\$ <FieldName>,<FieldLength>, <FieldType>,F \$ <FieldName>,<FieldLength>, <FieldType>,F...‡ Data message. The <FieldType> is C (character), N (numeric), \$(currency), D (date), or T (time). The F value is a placeholder for future use.

## Examples

### Agent Application to Agent Binary

```
AAGTListCallFields C COriginator_ID 11111 23 1$
list1‡
```

### Agent Binary to Agent Application

```
AGTListCallFields D Agent server 17939 23 42$
0 $
M00001 $
*
SYSNUM,4,N,F $ PRIN,4,C,F $CCODE,3,C,F $
ACCTNUM,16,N,F $ NAME,26,C,F $
NAME2,26,C,F $WFLAG,1,C,F $ PHONE2,10,N,F $
AREA2,3,N,F $ PHONE1,10,N,F $ AREA,3,N,F $
EXTERNAL,1,C,F $ INTERNAL,1,C,F $ BAL,10,$,F
$ CREDLINE,7,$,F $ DELQUENT,10,$,F $
DAYS,3,N,F $ PAYDAY,8,D,F $ PAYAMT,8,$,F $
ZIPCODE,5,N,F $ BEHSCORE,3,C,F $ AGENT,8,C,F
$ DTE,8,D,F $ TME,8,T,F $ CODE,2,C,F $
ENTRYDATE,8,D,F $ STATUSFLAG,1,C,F $
RECALLDATE,8,D,F $ RECALLTIME,8,T,F $
DAYSCNT,3,N,F $ PHONESTAT,2,C,F $
ZONEPHONE1,1,C,F $ ZONEPHONE2,1,C,F $
PHONECNT1,2,N,F $ PHONECNT2,2,N,F $
CURPHONE,2,N,F $ RECALLPHONE,2,C,F $
DUR4,9,N,F $ DUPE,1,C,F $ JOBNAME,20,C,F‡
AGTListCallFields R Agent server 17939 23 2$
```

0 §  
M00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### See also

[AGTListDataFields](#)

[AGTReadField](#)

[AGTSetDataField](#)

[AGTSetNotifyKeyField](#)

[AGTUpdateField](#)

## AGTListCallLists

AGTListCallLists is a system command.

#### Alternate name

ListCallLists

#### Function

Lists all calling lists on Proactive Contact. The command recognizes as a calling list any file with the fdict extension.

Use this command to collect calling list names for use with AGTListCallFields.

#### Availability

AGTListCallLists is available any time after AGTLogon executes.

#### Format

AGTListCallLists C <Client> <ProcID> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### Data parameters

None

**Return values**

Code	Description
M00000	Complete.
M00001	§ <ListName1> § <ListName2>...àData message. The data message includes as many data segments as there are calling lists on the system.

**Examples****Agent Application to Agent Binary**

```
AAGTListCallLists C COriginator_ID 11111 2 0‡
```

**Agent Binary to Agent Application**

```
AAGTListCallLists D Agent server 17939 2 7§
0 §
M00001 §
inbndl §
inbrpt §
latel §
list1 §
list2‡
AGTListCallLists R Agent server 17939 2 2§
0 §
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**See also**

[AGTListCallLists](#)

## AGTListDataFields

AGTListDataFields is a job setup command.

**Alternate name**

ListDataFields

**Function**

Lists the data fields in the calling list for the current job.



Use this command to:

- Identify field names to use with the AGTSetDataField, AGTSetNotifyKeyField, or AGTReadField commands
- Get field parameters to use with AGTUpdateField

The list type parameter controls whether AGTListDataFields expects outbound or inbound calling list fields. The parameter must match the type of calling list the attached job uses.

## Availability

The agent application must be attached to a job when sending this command. While attached to a job, AGTListDataFields can execute any time.

## Format

```
AAGTListDataFields C <Client> <ProclD> <InvokeID> 1$
                  <ListType> ‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## Data parameters

ListType	<p>Type of calling list used by the attached job: inbound (I) or outbound (O).</p> <p>Blend jobs use both I and O calling lists. Inbound jobs use type I calling lists. Outbound, Managed Dialing, Unit Work List, and Sales Verification jobs use type O calling lists.</p>
----------	--

## Return values

Code	Description
E28885	Agent application is not attached to a job. Execute AGTAttachJob and then retry.
E28891	Must specify I (inbound) or O (outbound) calling list type. Retry command with <ListType> parameter included.
E28892	No inbound calling list fields are available. Retry using <ListType> O.
E28893	No outbound calling list fields are available. Retry using <ListType> I.

Code	Description
M00000	Complete.
M00001	Data message. The <FieldType> is C (character), N (numeric), \$(currency), D (date), or T (time). The F value is a placeholder for future use.

## Examples

### Agent Application to Agent Binary

```
AGTListDataFields C COriginator_ID 11111 19 1$
O‡
```

### Agent Binary to Agent Application

```
AGTListDataFields D Agent server 15395 19 42$
0 $
M00001 $
*
SYSNUM,4,N,F $ PRIN,4,C,F $ CCODE,3,C,F $
ACCTNUM,16,N,F $ NAME,26,C,F $ NAME2,26,C,F
$WFLAG,1,C,F $ PHONE2,10,N,F $ AREA2,3,N,F $
PHONE1,10,N,F $ AREA,3,N,F $ EXTERNAL,1,C,F
$ INTERNAL,1,C,F $ BAL,10,$,F $ CREDLINE,7,$,F
$ DELQUENT,10,$,F $ DAYS,3,N,F $ PAYDAY,8,D,F
$ PAYAMT,8,$,F $ ZIPCODE,5,N,F $
BEHSCORE,3,C,F $ AGENT,8,C,F $ DTE,8,D,F $
TME,8,T,F $ CODE,2,C,F $ ENTRYDATE,8,D,F $
STATUSFLAG,1,C,F $ RECALLDATE,8,D,F $
RECALLTIME,8,T,F $ DAYSCNT,3,N,F $
PHONESTAT,2,C,F $ ZONEPHONE1,1,C,F $
ZONEPHONE2,1,C,F $ PHONECNT1,2,N,F $
PHONECNT2,2,N,F $ CURPHONE,2,N,F $
RECALLPHONE,2,C,F $ DUR4,9,N,F $ DUPE,1,C,F
$ JOBNAME,20,C,F‡
AGTListDataFields R Agent server 15395 19 2$
0 $
M00000‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## See also

[AGTAttachJob](#)

[AGTReadField](#)

[AGTSetDataField](#)

[AGTSetNotifyKeyField](#)[AGTUpdateField](#)

## AGTListJobs

AGTListJobs is a system command.

### Alternate name

ListJobs

### Function

Returns the job type, name, and status for each job defined on Proactive Contact.

Use AGTListJobs to identify choices for the AGTAttachJob command.

### Availability

This command is available any time after executing AGTLogon.

### Format

AGTListJobs C <Client> <ProcID> <InvokeID> 1§  
<JobType>†

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

JobType	Type of jobs to list: all (A), inbound (I), blend (B), Managed Dialing (M), or outbound (includes Unit Work List and Sales Verification) (O). Use one alphabetic, case sensitive, character.
Status	(I) inactive or (A) active

### Return values

Code	Description
E28884	The agent binary was unable to access Proactive Contact shared memory. This usually indicates an Proactive Contact system problem.
E70003	Unknown job type. Retry with a job type shown above.
M00000	Complete.

Code	Description
M00001	§ <JobType>, <JobName>, <Status> § <JobType>, <JobName>, <Status>...àData message containing requested job information. Job status appears as a single digit, I for inactive or A for active.

## Examples

### Agent Application to Agent Binary

```
AGTListJobs C COriginator_ID 11111 18 1$
A†
```

### Agent Binary to Agent Application

```
AGTListJobs D Agent server 15395 18 8 $
0 $
M00001 $
B,blend1,I $
I,inbnd1,I $
M,managed,I $
O,outbnd,I $
O,outbnd2,A $
O,outbndtest,I†
AGTListJobs R Agent server 15395 18 2$
0 $
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTAttachJob](#)

[AGTLogon](#)

## AGTListKeys

AGTListKeys is a system command.

### Alternate name

ListKeys

### Function

Returns call completion codes, descriptions, and their associated telephone script labels for the attached job.

Use this command to collect information for use in the AGTReleaseLine and AGTFinishedItem commands.

If the agent application does not execute AGTListKeys each time AGTAttachJob executes and store these return values, the AGTReleaseLine and AGTFinishedItem commands might return incorrect values to Proactive Contact.

**Availability**

Execute this command any time after executing AGTAttachJob.

**Format**

AGTListKeys C <Client> <ProclD> <InvokeID> 0 ‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None

**Return values**

Code	Description
E00518	Unable to open the keys file. Indicates an Proactive Contact system problem.
E28885	Not attached to a job. Execute AGTAttachJob and retry the command.
M00000	Complete.
M00001	Data message. Completion codes are two digits. Call centers define call completion codes during specification of their Proactive Contact system. The descriptions are from the Proactive Contact Compcode.cfg file. Telephone script labels are labels found in Proactive Contact Telephny.spt file. The system associates the labels with the respective call completion codes.

**Examples****Agent Application to Agent Binary**

```
AGTListKeys C COriginator_ID 11111 42 0‡
```

## Agent Binary to Agent Application

```
AGTListKeys D Agent server 1127 42 14$
0 $
M00001 $
35,Managed cancel call,cancel_call
89,Managed non-connection,call_complete
,*Record not yet called,pf_msg_1
,*Record not yet called,call_complete
,*Record not yet called,call_complete
,*Record not yet called,call_complete
19,Recall release,call_complete
16,Ringing phone,call_complete
17,Cust hung-up in queue,call_complete
,*Record not yet called,call_complete
6,,call_complete
8,,call_complete‡
AGTListKeys R Agent server 1127 42 2$
0 $ M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTAttachJob](#)

[AGTFinishedItem](#)

[AGTReleaseLine](#)

## AGTListScreens

AGTListScreens is a system command.

### Alternate name

ListScreens

### Function

Command message to get a list of screens of the specified type associated with the current job.

### Example

```
CLIENT --> AGTListScreens      COrigID      MoAgt 902    1^^O^C
SERVER <-- AGTListScreens      DAgent server 3455 902 3
^^0^^M00001^^list1^C
SERVER <-- AGTListScreens      RAgent server 3455 902 2    ^^0^^M00000^C
```

Here the client sends the command asking for outbound (O) screens and the server returns "list1". There are record separators between the names for multiple screens and the number of data fields (3 in the example) is one larger for each screen.

## AGTListState

AGTListState is a system command.

### Alternate name

ListState

### Function

Returns the agent's current state on Proactive Contact.

Use this command to determine the agent's current state before the agent application takes some action.

AGTListState can be particularly helpful before executing one of the commands that changes the agent state. For information on agent states and related commands, see [Agent States](#) on page 25.

### Availability

This command is available any time after AGTLogon executes.

### Format

AGTListState C <Client> <ProcID> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

None

### Return values

Code	Description
M00000	Complete.
S70000	, <JobName>Agent is on a call. Working on <JobName>, currently working with a customer record.
S70001	, <JobName>Agent is ready for a call. Working on <JobName>, currently waiting for next customer record.

Code	Description
S70002	, <JobName>Agent has joined (logged on to) a job. Working on <JobName>, currently available for work but not ready for next item.
S70003	, <JobName>Agent has selected a job to work with. Attached to <JobName>, but not yet available for work.
S70004	Agent logged on to Proactive Contact. Agent is idle, not yet attached to a job.

### Examples

#### Agent Application to Agent Binary

```
AGTListState C COriginator_ID 11111 61 0‡
```

#### Agent Binary to Agent Application

```
AGTListState D Agent server 5846 61 2$
0 $
S70003,managed1‡
AGTListState R Agent server 5846 61 2$
0 $
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### See also

[AGTAttachJob](#)

[AGTAvailWork](#)

[AGTFinishedItem](#)

[AGTNoFurtherWork](#)

[AGTReadyNextItems](#)

AGTListTenants is a system command.

#### Alternate name

ListTenants



**Function**

Lists the available tenants for agent. Use this command to identify the choices for AGTSetTenant.

**Availability**

Execute this command after agent login.

**Format**

AGTListTenants C <Client> <ProcID> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None

**Return values**

Code	Description
E58006	Failed to load tenant user mapping.
M00000	Complete.
M00001	§ <TenantName1>, <TenantName2>...‡ Data message. The data elements are the Tenant name values on the dialer.

**Examples****Agent Application to Agent Binary**

```
AGTListTenants C COriginator_ID 11111 12 0‡
```

**Agent Binary to Agent Application**

```
AGTListTenants D Agent server 4839 12 3§
0 §
M00001 §
Tenant1,Tenant2‡
AGTListTenants R Agent server 4839 12 2§
0 §
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)  
† = Message continues(ETB), ASCII x17  
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## AGTListUnits

AGTListUnits is a job setup command.

### Alternate name

ListUnits

### Function

Lists the available Unit ID values for a Unit Work List job.  
Use this command to identify choices for AGTSetUnit.  
Unit ID values correspond to the value in a specific field of the customer records.  
For example, if the Unit ID field is defined as LIMIT (the credit limit), AGTListUnits might return values of 2500, 5000, 7500, 10000. These values are credit limits that appear in customer records selected for the job. They are the Unit ID values available for the job.

### Availability

Execute this command after attaching a Unit Work List job with AGTAttachJob.

### Format

AGTListUnits C <Client> <ProcID> <InvokeID> 0‡  
§ = Message data separator, ASCII x1E (socket) or comma (tty)  
† = Message continues(ETB), ASCII x17  
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

None

### Return values

Code	Description
------	-------------

Code	Description
E28885	Not attached to a job. Attach an active Unit Work List job and retry.
E28886	The attached job is not a Unit Work List job. Change attached job and retry.
M00000	Complete.
M00001	§ <UnitID>, <UnitID>...‡ Data message. The data elements are the Unit ID values on the job.

## Examples

### Agent Application to Agent Binary

```
AGTListUnits C COriginator_ID 11111 12 0‡
```

### Agent Binary to Agent Application

```
AGTListUnits D Agent server 4839 12 3§
0 §
M00001 §
230,860‡
AGTListUnits R Agent server 4839 12 2§
0 §
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## See also

[AGTAttachJob](#)

[AGTSetUnit](#)

# AGTLogloStart

AGTLogloStart is a system command.

## Alternate name

LogloStart

**Function**

Creates a file on the Proactive Contact system named <AgentName>\_API.trans. It contains all the messages between the agent application and Proactive Contact.

Use this file to verify message content and to track the agent application session.

The Agent DLL automatically creates a message log in the directory containing the agent application. For information on the DLL log file, see [Understanding the Proactive Contact Agent API](#).

There is an entry in master.cfg: DEBUGDIR:\$ROOTDIR/debug

Any binary that specifies a DBGOUTPUT location that does not begin with a slash will get placed in that directory. If it begins with a slash, it will go into that absolute path location.

**Availability**

This command is available any time after AGTLogon executes.

**Format**

AGTLogIoStart C <Client> <ProcID> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII Data parameters

None

**Return values**

M00000	Complete.
--------	-----------

**Examples**

**Agent Application to Agent Binary**

AGTLogIoStart C COriginator\_ID 11111 26 0‡

**Agent Binary to Agent Application**

A

AGTLogIoStart R Agent server 5846 26 2§

0 §

M00000†

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Output File**

The log files replace the message data separator with ^ characters and use a carriage return to represent both the message continuation (ETB) symbol and the message termination (ETX) symbol.

**Outbound**

```
SERVER<-- AGTLogIoStart R Agent server 8617 9 2
^^0^^M00000
CLIENT--> AGTReserveHeadset C COriginator_ID 111111 2 1
,1
SERVER<-- AGTReserveHeadset P Agent server 8617 2 2
^^0^^S28833
SERVER<-- AGTReserveHeadset R Agent server 8617 2 2
^^0^^M00000
CLIENT--> AGTConnHeadset C COriginator_ID 111111 3 0
,
SERVER<-- AGTConnHeadset P Agent server 8617 3 2
^^0^^S28833
SERVER<-- AGTConnHeadset R Agent server 8617 3 2
^^0^^M00000
CLIENT--> AGTSetWorkClass C COriginator_ID 111111 4 1
,OUTBOUND
SERVER<-- AGTSetWorkClass R Agent server 8617 4 2
^^0^^M00000
CLIENT--> AGTAttachJob C COriginator_ID 111111 5 1
,blendgjjg
SERVER<-- AGTAttachJob R Agent server 8617 5 2
^^0^^M00000

CLIENT--> AGTAvailWork C COriginator_ID 111111 8 0
SERVER<-- AGTAvailWork P Agent server 8617 8 2
^^0^^S28833
SERVER<-- AGTAvailWork R Agent server 8617 8 2
^^0^^M00000
CLIENT--> AGTReadyNextItem C COriginator_ID 111111 7 0
SERVER<-- AGTReadyNextItem R Agent server 8617 7 2
^^0^^M00000
SERVER<-- AGTCallNotify N Agent server 8617 0 4
^^0^^M00001^^JOHN DOE^^OUTBOUND
SERVER<-- AGTCallNotify N Agent server 8617 0 2
^^0^^M00000
CLIENT--> AGTMoFlashBlind C COriginator_ID 111111 10
0
SERVER<-- AGTMoFlashBlind P Agent server 8617 10 2
^^0^^S28833
SERVER<-- AGTMoFlashBlind R Agent server 8617 10 2
^^0^^M00000
CLIENT--> AGTFinishedItem C COriginator_ID 111111 9 1
,20
SERVER<-- AGTFinishedItem R Agent server 8617 9 2
^^0^^M00000
CLIENT--> AGTReadyNextItem C COriginator_ID 111111 7 0
SERVER<-- AGTReadyNextItem R Agent server 8617 7 2
^^0^^M00000
SERVER<-- AGTCallNotify N Agent server 8617 0 4
^^0^^M00001^^JOHN DOE^^OUTBOUND
SERVER<-- AGTCallNotify N Agent server 8617 0 2
```

## Avaya Proactive Contact Agent API 5.2

```
^^0^^M00000
CLIENT--> AGTMoFlashSupv C COriginator_ID 111111 11
0
SERVER<-- AGTMoFlashSupv P Agent server 8617 11 2
^^0^^S28833
SERVER<-- AGTMoFlashSupv R Agent server 8617 11 2
^^0^^M00000
CLIENT--> AGTMoFlashSupv C COriginator_ID 111111 11
0
SERVER<-- AGTMoFlashSupv R Agent server 8617 11 2
^^0^^M00000
CLIENT--> AGTDumpData C COriginator_ID 111111 1 1
,D1
SERVER<-- AGTDumpData R Agent server 8617 1 2
^^0^^M00000

CLIENT--> AGTReleaseLine C COriginator_ID 111111 8
1 ,call_complete
SERVER<-- AGTReleaseLine P Agent server 8617 8 2
^^0^^S28833
SERVER<-- AGTReleaseLine R Agent server 8617 8 2
^^0^^M00000
CLIENT--> AGTDumpData C COriginator_ID 111111 1 1
,D1
SERVER<-- AGTDumpData R Agent server 8617 1 2
^^0^^M00000
CLIENT--> AGTFinishedItem C COriginator_ID 111111 9 1
,20
SERVER<-- AGTFinishedItem R Agent server 8617 9 2
^^0^^M00000
CLIENT--> AGTReadyNextItem C COriginator_ID 111111 7 0
SERVER<-- AGTReadyNextItem R Agent server 8617 7 2
^^0^^M00000
SERVER<-- AGTCallNotify N Agent server 8617 0 4
^^0^^M00001^^JOHN DOE^^OUTBOUND
SERVER<-- AGTCallNotify N Agent server 8617 0 2
^^0^^M00000
CLIENT--> AGTMoFlashSupv C COriginator_ID 111111 11
0
SERVER<-- AGTMoFlashSupv P Agent server 8617 11 2
^^0^^S28833
SERVER<-- AGTMoFlashSupv R Agent server 8617 11 2
^^0^^M00000
CLIENT--> AGTHangupCall C COriginator_ID 111111 9 0
SERVER<-- AGTHangupCall R Agent server 8617 9 2
^^0^^M00000
CLIENT--> AGTManualCall C COriginator_ID 111111 9
1 ,2068693245
SERVER<-- AGTManualCall R Agent server 8617 9 2
^^0^^M00000
CLIENT--> AGTReleaseLine C COriginator_ID 111111 8 1
,call_complete
SERVER<-- AGTReleaseLine P Agent server 8617 8 2
^^0^^S28833
SERVER<-- AGTReleaseLine R Agent server 8617 8 2
^^0^^M00000
CLIENT--> AGTFinishedItem C COriginator_ID 111111 9 1
,20
```

## Avaya Proactive Contact Agent API 5.2

```
SERVER<-- AGTFinishedItem R Agent server 8617 9 2
^^0^^M00000

CLIENT--> AGTReadyNextItem C COriginator_ID 111111 7 0
SERVER<-- AGTReadyNextItem R Agent server 8617 7 2
^^0^^M00000
SERVER<-- AGTCallNotify N Agent server 8617 0 4
^^0^^M00001^^JANE COE^^OUTBOUND
SERVER<-- AGTCallNotify N Agent server 8617 0 2
^^0^^M00000
CLIENT--> AGTMoFlashSupv C COriginator_ID 111111 11
0
SERVER<-- AGTMoFlashSupv P Agent server 8617 11 2
^^0^^S28833
SERVER<-- AGTMoFlashSupv R Agent server 8617 11 2
^^0^^M00000
CLIENT--> AGTReleaseLine C COriginator_ID 111111 8 1
,call_complete
SERVER<-- AGTReleaseLine P Agent server 8617 8 2
^^0^^S28833
SERVER<-- AGTReleaseLine R Agent server 8617 8 2
^^0^^M00000

Inbound
Following is an example of an output file.
SERVER<-- AGTLogIoStart R Agent server 8618 9 2
^^0^^M00000
CLIENT--> AGTReserveHeadset C COriginator_ID 111111 2 1
,2
SERVER<-- AGTReserveHeadset P Agent server 8618 2 2
^^0^^S28833
SERVER<-- AGTReserveHeadset R Agent server 8618 2 2
^^0^^M00000
CLIENT--> AGTConnHeadset C COriginator_ID 111111 3 0
,
SERVER<-- AGTConnHeadset P Agent server 8618 3 2
^^0^^S28833
SERVER<-- AGTConnHeadset R Agent server 8618 3 2
^^0^^M00000
CLIENT--> AGTSetWorkClass C COriginator_ID 111111 4 1
,I
SERVER<-- AGTSetWorkClass R Agent server 8618 4 2
^^0^^M00000

CLIENT--> AGTAttachJob C COriginator_ID 111111 5 1
,blendgjjg
SERVER<-- AGTAttachJob R Agent server 8618 5 2
^^0^^M00000
CLIENT--> AGTAvailWork C COriginator_ID 111111 8 0
SERVER<-- AGTAvailWork P Agent server 8618 8 2
^^0^^S28833
SERVER<-- AGTAvailWork R Agent server 8618 8 2
^^0^^M00000
CLIENT--> AGTReadyNextItem C COriginator_ID 111111 7 0
SERVER<-- AGTReadyNextItem R Agent server 8618 7 2
^^0^^M00000
SERVER<-- AGTCallNotify N Agent server 8618 0 4
```

## Avaya Proactive Contact Agent API 5.2

```
^^0^^M00001^^INBOUND CALL^^INBOUND
SERVER<-- AGTCallNotify N Agent server 8618 0 2
^^0^^M00000
CLIENT--> AGTFinishedItem C COriginator_ID 111111 9 1
,20
SERVER<-- AGTFinishedItem R Agent server 8618 9 2
^^0^^M00000
CLIENT--> AGTReadyNextItem C COriginator_ID 111111 7 0
SERVER<-- AGTReadyNextItem R Agent server 8618 7 2
^^0^^M00000
SERVER<-- AGTCallNotify N Agent server 8618 0 4
^^0^^M00001^^TRANSFER CALL^^INBOUND
SERVER<-- AGTCallNotify N Agent server 8618 0 2
^^0^^M00000
CLIENT--> AGTFinishedItem C COriginator_ID 111111 9 1
,20
SERVER<-- AGTFinishedItem R Agent server 8618 9 2
^^0^^M00000
CLIENT--> AGTReadyNextItem C COriginator_ID 111111 7 0
SERVER<-- AGTReadyNextItem R Agent server 8618 7 2
^^0^^M00000
SERVER<-- AGTCallNotify N Agent server 8618 0 4
^^0^^M00001^^TRANSFER CALL^^INBOUND
SERVER<-- AGTCallNotify N Agent server 8618 0 2
^^0^^M00000
CLIENT--> AGTReleaseLine C COriginator_ID 111111 8 1
,call_complete
SERVER<-- AGTReleaseLine P Agent server 8618 8 2
^^0^^S28833

SERVER<-- N Agent server 8618 0 2
^^1^^E70002,AGTReleaseLine_RESP(LNREL)
CLIENT--> AGTFinishedItem C COriginator_ID 111111 9 1
,20
SERVER<-- AGTFinishedItem R Agent server 8618 9 2
^^0^^M00000
CLIENT--> AGTReadyNextItem C COriginator_ID 111111 7 0
SERVER<-- AGTReadyNextItem R Agent server 8618 7 2
^^0^^M00000
SERVER<-- AGTJobEnd N Agent server 8618 0 2
^^0^^M00000
CLIENT--> AGTDetachJob C COriginator_ID 111111 32 0
SERVER<-- AGTDetachJob R Agent server 8618 32 2
^^0^^M00000
CLIENT--> AGTDisconnHeadset C COriginator_ID 111111 33 0
SERVER<-- AGTDisconnHeadset P Agent server 8618 33 2
^^0^^S28833
SERVER<-- AGTDisconnHeadset R Agent server 8618 33 2
^^0^^M00000
CLIENT--> AGTFreeHeadset C COriginator_ID 111111 34 0
SERVER<-- AGTFreeHeadset R Agent server 8618 34 2
^^0^^M00000
CLIENT--> AGTLogoff C COriginator_ID 111111 35 0
SERVER<-- AGTLogoff R Agent server 8618 35 2
^^0^^M00000
```



See also

[AGTDumpData](#)

[AGTLogIoStop](#)

AGTLogIoStop

AGTLogIoStop is a system command.

Alternate name

LogIoStop

Function

Stops the agent binary from writing to the <AgentName>\_API.trans file.  
Use this command to terminate the log file.

Availability

This command is available any time after AGTLogIoStart executes.

Format

AGTLogIoStop C <Client> <ProclD> <InvokeID> 0‡  
\$ = Message data separator, ASCII x1E (socket) or comma (tty)  
† = Message continues(ETB), ASCII x17  
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None

Return values

M00000	Complete.
--------	-----------

Examples

Agent Application to Agent Binary

AGTLogIoStop C COriginator\_ID 11111 65 0‡

Agent Binary to Agent Application

AGTLogIoStop R Agent server 5846 65 2\$  
0 \$  
M00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**See also**

[AGTLogloStart](#)

## AGTLogoff

AGTLogoff is a connection command.

**Alternate name**

Logoff

**Function**

Exits Proactive Contact. When AGTLogoff executes, the client-server session terminates.

**Availability**

To execute this command, the agent application must not be attached to a job. If the application is attached to a job, execute AGTDetachJob first.

**Format**

AGTLogoff C <Client> <ProcID> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None

**Return values**

Code	Description
E28916	There is a job attached. Detach the job and retry.
M00000	Complete.

Code	Description
S28833	Pending.

## Examples

### Agent Application to Agent Binary

```
AGTLogoff C COriginator_ID 11111 532 0‡
```

### Agent Binary to Agent Application

```
AGTLogoff R Agent server 5819 532 2§
0 §
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## See also

[AGTDetachJob](#)

[AGTLogon](#)

# AGTLogoffAcd

AGTLogoffAcd is a connection command.

## Alternate name

LogoffAcd

## Function

Removes the agent registration with Proactive Contact Dispatcher process. Removing the registration logs the agent out of the ACD and disconnects the agent's headset. After AGTLogoffAcd executes, Proactive Contact cannot acquire the agent.

Use this command to remove the agent from Agent Blending before logging the agent out of Proactive Contact.

After using this command to log out of Agent Blending and the ACD, the agent can continue to work as a non-Agent Blending agent.

If the agent chooses to continue working, the agent application executes AGTConnHeadset to establish a headset connection.

If the agent chooses to log out, the agent application executes AGTDetachJob, AGTFreeHeadset, and AGTLogoff.

### Availability

This command is only available when the agent is logged in to Agent Blending and the ACD.

### Format

AGTLogoffAcid C <Client> <Procid> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

None

### Return values

Code	Description
M00000	Complete.
S28833	Pending

### Examples

#### Agent Application to Agent Binary

```
AGTLogoffAcid C COriginator_ID 11111 122 0‡
```

#### Agent Binary to Agent Application

```
AGTLogoffAcid P Agent server 17939 122 2§  
0 §  
S28833‡  
AGTLogoffAcid R Agent server 17939 122 2§  
0 §  
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTConnHeadset](#)

[AGTDetachJob](#)

[AGTFreeHeadset](#)[AGTLogoff](#)[AGTLogoffAcd](#)[AGTReserveHeadset](#)

## AGTLogon

AGTLogon is a connection command.

### Alternate name

Logon

### Function

Verifies a user name and password on Proactive Contact. AGTLogon also registers with the Agent\_count binary that controls the number of agents logged onto the system.

### Availability

This command is the only command available to the agent application when it first connects to Proactive Contact. Execute this command before issuing any other commands.

### Format

AGTLogon C <Client> <ProcID> <InvokeID> 3  
§ <AgentName> § <Password> § <AgentAPIVersion>‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

AgentName	<p>Enter the user name of the calling agent. Use up to 19 characters, 7-bit USASCII character set. Do not include embedded spaces.</p> <p>Note: For multitenancy, the client can provide the tenant name with the agent name as follows :</p> <p>TenantName\AgentName</p> <p>Old agent client (prior to Avaya Proactive Contact 5.1) does not have the API support for setting the tenant, therefore on old agent client, you can specify the tenant name with agent name.</p>
-----------	--

Password	Enter the agent's password. Use two to eight characters, 7-bit USASCII character set. Do not include embedded spaces.
AgentAPIVersion	<p>The Version of Agent API being used by the client. The format should be:</p> <p>PCAPI_&lt;Majorversion&gt;.&lt;Minorversion&gt;.&lt;ServicePackVersion&gt;.&lt;Repackversion&gt;.&lt;Buildnumber&gt;</p> <p>Example: PCAPI_5.1.0.0.4</p> <p>Clients using newer versions of Agent API (starting from 5.1.0.0.4) will be provided with functionality introduced in those versions, whereas older clients won't be, thus maintaining backward compatibility.</p>

**Return values**

Code	Description
E28812	<AgentName> is logged on to another application. The agent might be logged on at another location. Retry logon with a different <AgentName>.
E28925	This agent application is already logged in to the system from this session. To change the <AgentName>, execute AGTLogoff and retry.
E28926	Invalid login. Indicates that the user name or password is invalid.
E50100	The maximum number of agents allowed by the Agent_count binary are already logged on to the system.
E70012	The agent current password is no longer valid. Enter a new password to continue the log in.
E58006	Failed to load tenant user mapping.
E58007	Unable to get tenant id for tenant name.
E58022	Tenant name is invalid or the user does not belong to the tenant.

Code	Description
M00000	Complete.
S28833	Pending.

## Examples

### Agent Application to Agent Binary

```
AGTLogon C COriginator_ID 11111 1 3$
01 $
01 $
PCAPI_5.1.0.0.4†
```

### Agent Binary to Agent Application

```
AGTLogon P Agent server 5819 1 2$
0 $
S28833‡
AGTLogon R Agent server 5819 1 2$
0 $
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17x03

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## See also

[AGTLogoff](#)

# AGTLogonAcd

AGTLogonAcd is a connection command.

## Alternate name

LogonAcd

## Function

Registers the agent with the Dispatcher process on Proactive Contact.

Use AGTLogonACD to log in an agent as an Agent Blending agent. Registering the agent logs in the agent to the ACD and connects the agent's headset.

Agents who register with the Dispatcher process are ACD agents and may only join outbound, Unit Work List, and Managed Dialing jobs.

**Availability**

This command is available on systems with Agent Blending. Execute this command after reserving a headset and attaching a job. Do not execute AGTConnHeadset before executing this command.

**Format**

```
AGTLogonAcd C <Client> <ProclD> <InvokeID> 2§
               <Extension> § <PBX ID>†‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

Extension	The ACD extension to associate with the agent's telephone. Use a numeric value, up to 20 characters.
PBX ID	The identification of the PBX (switch) used for Agent Blending. Default value is 1. Value given here must match the DBKGROUP parameter in the Proactive Contact master.cfg file.

**Return values**

Code	Description
E28872	The agent is already logged on to ACD.
E28873	Headset is not reserved. Reserve a headset identification with AGTReserveHeadset before executing this command.
E28950	The specified extension is not one of the ACD extensions defined on Proactive Contact.
E28951	The specified extension is in use by another agent.
E28952	Cannot add another agent. The maximum number of ACD agents are already logged on.
E28953	Could not read the file containing the ACD extensions. Indicates an Proactive Contact system problem.



Code	Description
E28954	Duplicate logon - Please try again. Indicates that the agent user name is already logged in.
E28955	The Agent Blending Dispatcher process is not running.
E28956	ACD logon error. Dispatcher returned an “unknown” response to the agent application.
E70011	Agent Blending is not available on this Proactive Contact system. This system cannot work with ACD agents.
M00000	Complete.
S28833	Pending.

## Examples

### Agent Application to Agent Binary

```
AGTLogonACD C COriginator_ID 11111 7 2$
2315 $
1‡
```

### Agent Binary to Agent Application

```
AGTLogonACD P Agent server 17939 7 2$
0 $
S28833‡
AGTLogonACD R Agent server 17939 7 2$
0 $
M00000‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTLogoffAcd](#)

## AGTManagedCall

AGTManagedCall is a working command.

**Alternate name**

ManagedCall

**Function**

Used on Managed Dialing jobs to place a call to the customer before the preview period elapses.

If the preview period elapses without the agent releasing the customer record (see AGTFinishedItem), the agent binary executes AGTManagedCall automatically.

**Format**

AGTManagedCall C <Client> <ProcID> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None

**Return values**

Code	Description
E28885	Not attached to a job. Attach a job and retry command.
E28900	§ <message>‡ An Proactive Contact system internal error occurred. Error message text follows.
E28901	Agent is not available for work. Retry after executing AGTAvailWork.
E28907	Attached job is not a Managed Dialing job. Command is only available for Managed Dialing jobs.
E28908	Agent is not previewing a customer record. Retry after receiving a record to preview.
E28909	Managed call is already complete. The AGTManagedCall command is already being executed. Might occur when agent executes the command just as the time-out elapses.

Code	Description
E28910	Managed call already canceled or complete. Either an AGTFinishedItem, AGTReleaseLine or AGTManagedCall command is already in the process of executing. Might occur if agent mistakenly executes a line release rather than a finished item or if the finished item executes just as the time-out elapses.
E28911	Managed call already canceled. AGTFinishedItem is already executing.
E28964	Agent phone is busy. Release phone line before proceeding.
E28965	Softdialer link is down. Please wait until the link is up.
M00000	Complete.
M00001	§ <CallStatus>àData message. <CallStatus> should always be (CONNECT).
S28833	Pending. Call is being placed.

## Examples

### Agent Application to Agent Binary

```
AGTManagedCall C COriginator_ID 11111 1 0†
```

### Agent Binary to Agent Application

```
AGTManagedCall P Agent server 2570 1 2§
0 §
S28833†
AGTManagedCall D Agent server 2570 1 3§
0 §
M00001 §
(CONNECT)‡
AGTManagedCall R Agent server 2570 1 2§
0 §
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTFinishedItem](#)

[AGTPreviewRecord](#)

[AGTReadyNextItem](#)

## AGTManCallAnswered

AGTManCallAnswered is a notification event that notifies when the manual call is answered in managed job.

### Alternate name

ManCallAnswered

### Function

Sent when customer answers the call in manual call in managed job (VISUAL\_CPA is off).

### Availability

This notification is available on managed jobs and the MANCALLANSWER\_NOTIFY flag is set to true and VISUAL\_CPA is off in master.cfg

### Format

AGTManCallAnswered NAgent server 8978 0 2 § 0 §M00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

None

### Return Values

Code	Description
M00000	Complete

## AGTManualCall

AGTManualCall is a working command.

### Alternate name

ManualCall

**Function**

Places a manual call.

Use this command to call another agent, a supervisor, or an outside number. Do not use this command to transfer a call or establish a conference call. See [AGTTransferCall](#) and [AGTHookflashLine](#).

To hang up a call and keep the line open without placing a manual call, execute AGTHangupCall.

**Availability**

This command is available when the agent is working with a customer record and has an open telephone line. An agent gets an open telephone line by receiving a call from a customer.

If the open telephone line is connected to a customer, AGTHangupCall executes before the system places the new call.

This command is not available on an Avaya Proactive Contact with CTI system.

**Format**

```
AGTManualCall C <Client> <ProcID> <InvokeID> 1
                § <PhoneNumber> ‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

‡ = Message continues(ETB), ASCII x17

PhoneNumber	The phone number to call. It must be numeric, up to 43 characters, no embedded spaces or special characters. If the number is a complete telephone number rather than an extension, the format must match the standard format in the phonefmt.cfg configuration file on the Proactive Contact system.
-------------	---

**Return values**

Code	Description
E28843	§ <PhoneNumber>‡ The <PhoneNumber> is invalid. Either the format of the number does not match the standard phone format on Proactive Contact or the telephone number itself is not valid.
E28866	Telephone line is not available. Proactive Contact cannot place an outbound call. Retry after receiving a call from a customer.

Code	Description
E28967	An agent attempted to log out of an agent-owned recall without logging out of the job from which the agent transferred.
E29950	This feature is not available on an Proactive Contact with CTI system.
M00000	Complete.
S28814	Transfer in progress. The last call is still in the process of being transferred. Retry after transfer complete.

### Examples

#### Agent Application to Agent Binary

```
AGTManualCall C COriginator_ID 11111 1 1$
2068693245‡
```

#### Agent Binary to Agent Application

```
AGTManualCall R Agent server 2570 1 2$
0 $
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### See also

[AGTDialDigit](#)

[AGTHangupCall](#)

[AGTHookflashLine](#)

[AGTTransferCall](#)

## AGTManualPhone

AGTManualPhone is a working command.

#### Alternate name

ManualPhone

**Function**

Using this Command the Agent application will provide a phone number to the dialer and the dialer will return the formatted phone number to the agent application for dialing through a third party application.

The format of phone number will depend upon a new parameter THIRDPARTY\_PHONEFMT in master.cfg.

THIRDPARTY\_PHONEFMT:dialer

If the value of this parameter is "dialer" then the dialer will follow the dialer's phone format rule.

THIRDPARTY\_PHONEFMT:+

If the value is other than "Dialer" then

- The dialer will generate the full phone number including country code.
- Remove any non numeric digit if present in the phone number
- Finally if any value is specified in the THIRDPARTY\_PHONEFMT parameter then it will be prepended to the phone number and sent to the agent application.

**Availability**

This command is available when the agent is working with a customer record and has an open telephone line.

**Format**

AGTManualPhone C <Client> <ProcID> <InvokeID> 1  
§ <PhoneNumber> ‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

PhoneNumber	The phone number to call. It must be numeric, up to 43 characters, no embedded spaces or special characters. If the number is a complete telephone number rather than an extension, the format must match the standard format in the phonefmt.cfg configuration file on the Proactive Contact system.
-------------	---

**Return Values**

Code	Description
------	-------------

Code	Description
E28843	§ <PhoneNumber>† The <PhoneNumber> is invalid. Either the format of the number does not match the standard phone format on Proactive Contact or the telephone number itself is not valid.
M00000	Complete.
M00001	Data message.

## Examples

### Agent Application to Agent Binary

```
AGTManualPhone C COriginator_ID 11111 1 1$
2068693245‡
```

### Agent Binary to Agent Application

```
AGTManualPhone D Agent server 10668 111 3$
0$
M00001$
+12068693245‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## AGTMoFlashBlind

AGTMoFlashBlind is a working command.

### Alternate name

MoFlashBlind

### Function

Transfers a customer call and the record to another job. This command disconnects the transferring agent before transferring the call and customer record. Proactive Contact places the customer call in the queue for the new job or connects the customer to an available agent.

If Proactive Contact uses ANI/DNIS information, the transferred call carries the outbound number Proactive Contact dials as the ANI and the original outbound job



name as the DNIS. The inbound or blend job receiving the call sees the call as an inbound call.

Use this command to transfer any information collected by the agent during the conversation.

Keep the following limitations in mind:

- An agent cannot transfer a call that he or she received as a transfer call.
- An agent can transfer a call to a blend job with no inbound lines.
- An agent cannot cancel a blind transfer once it begins. (When the AGTMoFlashBlind command begins to execute, Proactive Contact disconnects the agent from the call. The agent disconnect is just as if the agent executed AGTReleaseLine.)
- If an agent transfers the call to the blend job on which he or she is working, it is possible that the inbound call will route back to the same agent. In that case the agent will not be able to transfer the call again.
- AGTMoFlashBlind cannot establish a conference call.

Use AGTMoFlashSupv to keep the agent connected while the transfer completes or to transfer the customer record and establish a conference call.

Use AGTHookflashLine or AGTTransferCall to transfer the telephone call but not the customer record.

The following tables show all possible transfers.

FROM	
Job Type	Agent Type
Outbound	Outbound
Outbound	Outbound
Blend	Outbound
Blend	Blend
Managed	Managed
Managed	Managed

TO	
----	--

<b>TO</b>	
Job Type	Agent Type
Inbound	Inbound
Blend	Blend or Inbound
Inbound	Inbound
Blend	Blend or Inbound
Inbound	Inbound
Blend	Blend or Inbound

The transferring agent works with a customer record which is part of the outbound calling list. When the call is transferred to an inbound or blend agent, the receiving agent works with a customer record which is part of the inbound calling list. Each agent's customer record updates occur independently.

### Availability

This command is available only for outbound calls on Proactive Contact systems configured for voice and data transfer.

This command is available when an agent is working with a customer record and is on the phone with the customer.

### Format

```
AGTMoFlashBlind C <Client> <ProcID> <InvokeID> 1§
                <JobName>‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

JobName	The value is the name of an active inbound or blend job on Proactive Contact that will receive the call. Use up to 19 characters, 7-bit USASCII. The <JobName> is case sensitive and may not include special characters or embedded spaces. This parameter is optional if the Proactive Contact *.job file for the original job contains a default transfer job name.
---------	---

**Return values**

Code	Description
E28866	No phone line is available. Proactive Contact cannot place the transfer call.
E28867	Line is not off-hook. There is no telephone call to transfer.
E28942	Transfer job is not available. The <JobName> specified is not running, does not exist, or is not an inbound or blend job.
E28985	This error code appears when the customer is not online. This can also occur when the customer hangup the line or the line is released.
E70007	Cannot transfer an inbound call. AGTMoFlashBlind can only transfer outbound calls.
E70008	Must specify a transfer job. There is no default transfer job configured on Proactive Contact, so you must include a transfer <JobName> with the command.
E70009	,<process name> Unable to send a message to <process name>. Internal Proactive Contact error; agent cannot send messages.
M00000	Complete.
S28814	Trunk-to-trunk transfer is already in progress. Might mean the agent previously executed AGTTransferCall.
S28833	Pending

## Examples

### Agent Application to Agent Binary

```
AGTMoFlashBlind C COriginator_ID 11111 129 1$  
outbnd1+
```

### Agent Binary to Agent Application

```
AGTMoFlashBlind P Agent server 8735 129 1$  
S28833+  
AGTMoFlashBlind R Agent server 8735 129 2$  
0 $  
M00000+
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTHookflashLine](#)

[AGTMoFlashSupv](#)

[AGTTransferCall](#)

## AGTMoFlashSupv

AGTMoFlashSupv is a working command.

### Alternate name

MoFlashSupv

### Function

Transfers a telephone call and the customer record to another job, maintaining the transferring agent's telephone connection. If Proactive Contact uses ANI/DNIS information, the transferred call carries the outbound number Proactive Contact dials as the ANI and the original outbound job name as the DNIS. The inbound or blend job receiving the call sees it as an inbound call.

The transferring agent stays connected to the telephone call during the transfer. The system places the customer on hold. If no agent is available to take the call, the agent binary returns an error and Proactive Contact reconnects the client with the transferring agent.

An agent cannot transfer a call that is received as a transfer call.

### Availability

This command is available only for outbound calls on Proactive Contact systems configured for voice and data transfer.

This command is available when an agent is working with a customer record and is on the phone with the customer.

The following tables show all possible transfers.

FROM	
Job Type	Agent Type
Outbound	Outbound
Outbound	Outbound
Blend	Outbound
Blend	Blend
Managed	Managed
Managed	Managed

TO	
Job Type	Agent Type
Inbound	Inbound
Blend	Blend or Inbound
Inbound	Inbound
Blend	Blend or Inbound
Inbound	Inbound
Blend	Blend or Inbound

The transferring agent works with a customer record which is part of the outbound calling list. When the customer call transfers to an inbound or blend agent, the receiving agent works with a customer record which is part of the inbound calling list. Each agent's customer record updates occur independently.

### Executing transfers

When Proactive Contact begins executing AGTMoFlashSupv, it places the customer on hold. In the simplest case, Proactive Contact connects the agent transferring the call with the receiving agent. The agents speak with each other while the customer remains on hold. The customer's record appears on each agent's screen. After conferring with the receiving agent, the transferring agent can do one of the following:

Execute AGTMoFlashSupv a second time to initiate a three-way conference call

Execute AGTReleaseLine or AGTFinishedItem to disconnect from the call and connect the customer to the receiving agent

Either the agent transferring the call or the agent receiving the call can cancel the transfer. Cancellations occur after Proactive Contact connects the agents.

If the transferring agent wishes to cancel the transfer, the transferring agent executes AGTHangupCall. This causes Proactive Contact to hang up the transfer and reconnect the customer to the transferring agent. The receiving agent must then complete the inbound call by releasing the line and the customer record.

If the receiving agent wishes to cancel the transfer, the receiving agent executes AGTReleaseLine or AGTFinishedItem. This causes Proactive Contact to disconnect the receiving agent and reconnect the customer to the transferring agent.

Use AGTTransferCall or AGTHookflashLine to transfer only the telephone call.

### Format

```
AGTMoFlashSupv C <Client> <ProcID> <InvokeID> 1§  
                <JobName>‡
```

```
AGTMoFlashSupv C <Client> <ProcID> <InvokeID> 1§
```

‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

JobName	The value is the name of an active inbound or blend job on Proactive Contact to receive the transferred call. Use up to 19 characters, 7-bit USASCII. The <JobName> is case sensitive and might not include special characters or embedded spaces. This parameter is optional if the Proactive Contact *.job file for the original job contains a default transfer job name.
---------	--

**Return values**

Code	Description
E28628	Transfer failed - try again later.
E28866	No phone line is available. Proactive Contact does not have a telephone line available to place the transfer call.
E28867	Line is not off-hook. There is no call to transfer.
E28868	The transfer failed. Try again later. There is no agent available on the specified job to take the transfer call.
E28942	Transfer job is not available. The <JobName> specified does not exist, is not active, or is an inbound or blend job.
E28985	This error code appears when the customer is not online. This can also occur when the customer hangup the line or the line is released.
E70007	Cannot transfer an inbound call. AGTMoFlashSupv can only transfer outbound calls.
E70008	Must specify a transfer job. There is no default transfer job configured on Proactive Contact. You must include a transfer <JobName> with the command.
E70009	,<process name> Unable to send a message to <process name>. Internal Proactive Contact error; agent cannot send messages.
E70010	Conference is in progress. The agent executed the command a third time after the conference call began.
M00000	Complete.

Code	Description
S28814	Trunk-to-trunk transfer is already in progress. Might mean the agent previously executed AGTTransferCall.
S28833	Pending.

## Examples

### Agent Application to Agent Binary

```
AGTMoFlashSupv C COriginator_ID 11111 129 1$
outbndl†
```

### Agent Binary to Agent Application

```
AGTMoFlashSupv P Agent server 8735 129 1$
S28833‡
AGTMoFlashSupv R Agent server 8735 129 2$
0 $
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## See also

[AGTHookflashLine](#)

[AGTMoFlashSupv](#)

[AGTTransferCall](#)

# AGTNoFurtherWork

AGTNoFurtherWork is a working command.

## Alternate name

NoFurtherWork

## Function

Removes the agent from being available for work on the current job. This is the agent logout of an active job.

Executing this command is no guarantee that the agent will not receive additional calls or preview messages. The agent might receive one additional call or a preview message while the AGTNoFurtherWork command is pending. If the agent is the last



agent to log out of a job, Proactive Contact keeps the logout in a pending state until the calls in progress for the job are complete.

Until the agent application receives the completion message from agent, the agent is not logged out of the job.

After executing AGTNoFurtherWork, the job is still attached to the agent application. To allow the agent to log in to another job, the application must first detach the current job and attach the new job.

**Availability**

This command might execute any time the agent is available for work.

If the agent is currently working with a customer record or a call, Proactive Contact holds the request until AGTFinishedItem executes.

**Format**

AGTNoFurtherWork C <Client> <ProclD> <InvokelD> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None

**Return values**

Code	Description
E28917	No job is attached. There is nothing to log out.
E28918	Not available for work on the job. Agent is not logged in to a job; nothing to log off.
E28967	An agent attempted to log out of an agent-owned recall without logging out of the job from which the agent transferred.
M00000	Complete.
S28833	Pending. This state can last through one or more customer calls.

## Examples

### Agent Application to Agent Binary

```
AGTNoFurtherWork C COriginator_ID 11111 1 0+
```

### Agent Binary to Agent Application

```
AGTNoFurtherWork P Agent server 28705 1 2$
  0 $
  S28833‡
AGTNoFurtherWork R Agent server 28705 1 2$
  0 $
  M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTAttachJob](#)

[AGTAvailWork](#)

[AGTDetachJob](#)

[AGTFinishedItem](#)

## AGTPreviewRecord

AGTPreviewRecord is a notification event message from the server.

### Alternate name

PreviewRecord

### Function

Notifies the agent application that Proactive Contact is transferring a customer record to the agent for preview.

Use AGTPreviewRecord during a Managed Dialing job to gather information to display on the agent screen.

The agent binary sends at least two data messages to the agent application for each preview event.

The first message contains any agent information and the call type (always Managed). If there is a key field for the current job, the key field name and key field data are part of this message.

Agent does not send a second data message if the agent or the agent application has not requested additional information by executing AGTListDataFields.

If the agent or the agent application requests multiple fields, AGTPreviewRecord sends the field names and values in the order the AGTSetDataField commands executed.

The final message is always a completion message.

To cancel the Managed call, execute AGTFinishedItem before the preview period expires.

### Availability

The event occurs when an agent is attached to a Managed Dialing job, available for work, and ready for the next customer record.

If the agent binary receives an unexpected preview event when the agent state is not appropriate to receive a call, error messages might occur instead of the AGTPreviewRecord data messages.

### Format

```
AGTPreviewRecord N <Agentserver> <ProcID> <InvokeID> 6§
    0 § M00001 §<OpMesg> § <CallType>
    §<NotifyFieldName>, <FieldData>‡
```

```
AGTPreviewRecord N <Agentserver> <ProcID> <InvokeID> 3§
    0 §M00001 § <FieldName>,<FieldData> §
    <FieldName>,<FieldData>...‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

‡ = Message continues(ETB), ASCII x17

OpMesg	<p>Information for the agent from preview record. The message is field information from the customer record. The default field used is NAME. Alljobs.dat on the Proactive Contact system defines the field. The text string "(Preview)" always follows the information. If the call is an Agent Owned Recall, the agent binary notifies the agent of a pending recall at the increment set by the RECALL_NOTIFY entry in the job file.</p> <p>The pending notification is:</p> <p>"AOR: &lt;Customer Name&gt; J: &lt;job name&gt; U: &lt;user ID&gt; T: HH.MM.SS." . In this case the &lt;OpMesg&gt; contains the field name present in Alljobs.dat and the pending notification separated by pipe( ).</p> <p>Example:</p> <p>Home phone - 2037538811 (Preview)   AOR: JOHN DOE Job: managed Unit: Allid Time: 16.50.00</p> <p>The &lt;OpMesg&gt; for the recall is "Agent &lt;Agent ID&gt; Owned Recall (Preview)"</p>
CallType	<p>Always MANAGED. Only Managed Dialing jobs send preview records.</p>

NotifyFieldName	Field name from the calling list which the agent application requested by executing AGTSetNotifyKeyField.
NotifyFieldData	Value of notification key field from the customer record.
FieldName	A field name from the calling list which the agent application requested by executing AGTSetDataField.
FieldData	Value of data field from the customer record.

**Return values**

Code	Description
E28885	The agent application is not attached to a job. Preview record event is inappropriate.
E28900	§ <message>‡ An Proactive Contact system internal error occurred. Error message text follows.
E28906	The agent is not ready for next customer record. Preview record event is inappropriate.
M00000	Complete.
M00001	§ <OpMesg> § <CallType> § <NotifyFieldName>, <FieldData>àInitial preview data message.
M00001	§ <FieldName>,<FieldData> § <FieldName>,<FieldData>...àAdditional preview field data message.

**Examples****Agent Binary to Agent Application**

```

AGTPreviewRecord N Agent server 17970 0 5$
0 $
M00001 §
PHONE1,2037478754
NAME, JOHN DOE
ACCTNUM,4302209860046261‡
MANAGED
AGTPreviewRecord N Agent server 17970 0 3$

```

```
0 $  
M00001 $  
BAL,534†  
AGTPreviewRecord N Agent server 17970 0 2$  
0 $  
M00000†
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### See also

[AGTManagedCall](#)

[AGTReadyNextItem](#)

[AGTSetDataField](#)

[AGTSetNotifyKeyField](#)

## AGTReadField

AGTReadField is a working command.

#### Alternate name

ReadField

#### Function

Gets a data field format and the value in the current customer record from the calling list.

Use this command to verify format and the value to change before updating a customer record.

The application might read any known field in the outbound or inbound calling list used with the job.

To get the formats for all calling list fields, use AGTListDataFields.

#### Availability

This command is available when the agent is working with a customer record. It does not require a telephone line.

#### Format

```
AGTReadField C <Client> <ProcID> <InvokeID> 2$  
                <ListType> § <FieldName>‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

ListType	Type of calling list the attached job uses: inbound (I) or outbound (O). Blend jobs use both I and O calling lists. Inbound jobs use type I calling lists. Outbound, Managed Dialing, Unit Work List, and Sales Verification jobs use type O calling lists.
FieldName	A field name from the calling list, as shown by AGTListDataFields. Up to 19 characters, alphanumeric, case sensitive, no embedded spaces.

**Return values**

Code	Description
E28885	Not attached to a job. Retry with an active job and an open customer record.
E28891	Must specify inbound or outbound operation. Retry with <ListType> set to I or O.
E28892	No inbound calling list fields available. Retry with <ListType> set to O.
E28893	No outbound calling list fields available. Retry with <ListType> set to I.
E28894	<FieldName> not found in calling list. Retry with valid field name for the calling list.
E28912	Customer record is not available. The agent is not working with a customer record.
M00000	Complete.
M00001	\$ <FieldName>,<FieldType>, <FieldLength>,<FieldValue> àData message. <FieldName> is the name specified in the command. <FieldType> is the type of data in the field: A (alphanumeric), N (numeric), D (date), T (time), or \$ (currency). <FieldLength> is the number of characters in the field. <FieldValue> is the value in the current customer record.

**Examples****Agent Application to Agent Binary**

```
AGTPreviewRecord N <Agentserver> <ProcID> <InvokeID> 3$
0 $
M00001 $
<FieldName>,<FieldData> $
```

<FieldName>,<FieldData>...‡

### Agent Binary to Agent Application

```
AGTReadField D Agent server 2570 1 3$
0 $
M00001 $
PHONE2,N,10,0000000000‡
AGTReadField R Agent server 2570 1 2$
0 $
M00000‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTListDataFields](#)

[AGTUpdateField](#)

## AGTReadyNextItem

AGTReadyNextItem is a working command.

### Alternate name

ReadyNextItem

### Function

Permits Proactive Contact to deliver a call or a customer record for preview to the agent application.

Use this command after the completion of every call or record preview and immediately following initial login to a job. See [AGTAvailWork](#).

### Availability

The command is available when the agent is available for work.

A disconnected headset or pending AGTNoFurtherWork disables the command.

### Format

```
AGTReadyNextItem C <Client> <ProcID> <InvokeID> 0‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None

**Return values**

Code	Description
E28885	Not attached to a job. Attach a job, become available for work, and retry.
E28896	Headset must be active. The agent's headset is disconnected. Reconnect the headset and retry.
E28901	Not available for work. Execute AGTAvailWork and retry.
E28902	Already on a customer record. Release the current record with AGTReleaseLine or AGTFinishedItem and retry.
E28903	Already set ready for next customer record. AGTReadyNextItem already executed.
E28904	Request for no further work pending. Command is disabled.
E28905	Request to transfer to another job is active. Retry after job transfer is complete.
E28964	Agent phone is busy. Release phone line before proceeding.
E28965	Softdialer link is down. Please wait until the link is up.
M00000	Complete.

**Examples****Agent Application to Agent Binary**

```
AGTReadyNextItem C COriginator_ID 11111 1 0†
```

**Agent Binary to Agent Application**

```
AGTReadyNextItem R Agent server 29722 1 2$
0 $
M00000†
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17



‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**See also**

[AGTAvailWork](#)

[AGTFinishedItem](#)

[AGTManagedCall](#)

[AGTNoFurtherWork](#)

[AGTPreviewRecord](#)

[AGTReleaseLine](#)

## AGTReceiveMessage

AGTReceiveMessage is a notification event message from the server.

**Alternate name**

ReceiveMessage

**Function**

Forwards a text message sent from the Proactive Contact supervisor to the agent. Messages sent this way require one line of screen space. The agent application displays the contents of the data message on the agent's screen.

**Availability**

This event can occur any time after AGTLogon.

An error message posts with this event name only when an Proactive Contact internal error occurs.

**Format**

AGTReceiveMessage N <Agentserver> <ProcID> <InvokeID> 3§  
0 § M00001 § <Message>‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

Message	The text of the message from the supervisor. The message can be up to 79 characters long. Embedded spaces and special characters are allowed.
---------	---

**Return values**

Code	Description
E28900	§ <message>‡ An Proactive Contact system internal error occurred. Error message text follows.
M00000	Complete.
M00001	§ <Message>àData message. The text of the message the supervisor sent.

**Examples****Agent Binary to Agent Application**

```

AGTReceiveMessage N Agent server 1268 0 3$
0 $
M00001 $
Please call Supervisor‡
AGTReceiveMessage N Agent server 1268 0 2$
0 $
M00000‡

```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

‡ = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**See also**

[AGTSendMessage](#)

## AGTJobMode

AGTJobMode is a notification event message from the server.

**Alternate name**

JobMode

**Function**

Notifies the agent application that the agent has joined a manual mode or Preview empty record enabled job.

In case of a Manual mode enabled job the client will receive S28996 code along with flags for manual mode and click to dial.

In case of a Preview empty record enabled job the client will receive S28997 code.

**Availability**

This event occurs when an agent joins a manual mode enabled job or the Preview empty record feature is enabled in a Job.

**Format**

In case of Manual Mode enabled job:

AGTJobMode N <Agent server> <ProcID> <InvokeID> 4 § 0 § <Message>‡

In case of Preview Empty record enabled job:

AGTJobMode N <Agent server> <ProcID> <InvokeID> 2 § 0 § <Message>‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

‡ = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

Message	The Proactive Contact message number and any data or information that is part of that message.
---------	--

**Return values**

Code	Description
S28996	§ <Message>‡ This message is for Manual Mode enabled job. This will be followed by extra information i.e. two flags set in the job. The first flag specifies whether the job is manual mode enabled or not and the second flag will give information about ClickToDial feature is enabled or not.
S28997	§ <Message>‡ This message indicates that Preview empty record feature is enabled in a Job.

**Examples****Agent Binary to Agent Application**

```
AGTJobMode N Agent server 3916 0 4§  
0 §  
S28996§
```

```
1$  
1†  
AGTJobMode N Agent server 24046 0 2$  
0 $  
S28997‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## AGTReleaseLine

AGTReleaseLine is a working command.

### Alternate name

ReleaseLine

### Function

Allows the agent to continue updating the customer record after the conversation is over. (To release the customer record, execute AGTFinishedItem.)

Use AGTReleaseLine to release the telephone line separately from the customer record or to cancel a Managed Dialing call.

There are two data parameters: <ScriptLabel> and <MessageNo>.

The <ScriptLabel> parameter refers to a label in the /usr/vl/scripts/ telephny.spt file.

The <MessageNo> parameter refers to a message number configured in the /usr/vl/config/voicemsg.cfg file.

Proactive Contact expects to receive at most one of these parameters, not both. If the agent application provides both parameters, Proactive Contact uses the <ScriptLabel> parameter but not the <MessageNo> parameter.

- If the agent application provides the <ScriptLabel> parameter, Proactive Contact executes the matching label in the telephny.spt script file and releases the line.
- If the agent application provides the <MessageNo> parameter, Proactive Contact releases the telephone line after playing the message indicated by the message number to the customer.
- If the agent application does not provide either parameter, Proactive Contact releases the telephone line without executing a script or playing a message to the customer.

If the agent application requests AGTFinishedItem without requesting AGTReleaseLine first, AGTFinishedItem executes AGTReleaseLine with no parameters.

### Availability

The command is available when the agent is working with a customer record and is talking with the customer.

**Format**

AGTReleaseLine C <Client> <ProcID> <InvokeID> 0‡

AGTReleaseLine C <Client> <ProcID> <InvokeID> 1§  
                   <ScriptLabel>‡

AGTReleaseLine C <Client> <ProcID> <InvokeID> 2§  
                   §  
                   <MessageNo>‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

‡ = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

ScriptLabel	Optional parameter. Value is the script label to execute in the calling script (telephny.spt). Use AGTListKeys to get available labels. Use up to 15 characters, 7-bit USASCII case sensitive text with no embedded spaces or special characters.
MessageNo	Optional parameter. Value is the message number to play to the customer. These numbers must appear in the voicemsg.cfg file on the Proactive Contact system. Use up to three digits, ranging from 1 to 255.

**Return values**

Code	Description
E28866	Failed to release the line for the Managed Dialing preview (no telephone line available). This message results when Proactive Contact is in the process of placing the preview call.
E28885	Not attached to a job. Retry when attached to a job, available for work, and working with a customer record.
M00000	Complete.
S28833	Pending.

## Examples

### Agent Application to Agent Binary

```
AGTReleaseLine C COriginator_ID 11111 47 2$  
    call_complete‡
```

### Agent Binary to Agent Application

```
AGTReleaseLine P Agent server 6111 47 2$  
    0 $  
    S28833‡  
AGTReleaseLine R Agent server 6111 47 2$  
    0 $  
    M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTFinishedItem](#)

[AGTListKeys](#)

## AGTReserveHeadset

AGTReserveHeadset is a connection command.

### Alternate name

ReserveHeadset

### Function

Identifies a headset and reserves memory for a headset connection. Proactive Contact can only reserve one headset per user name.

Use this command to establish the relationship between the agent application workstation and a telephone extension.

### Availability

Execute this command after executing the AGTLogon command and before executing the AGTConnHeadset command.

### Format

```
AGTReserveHeadset C <Client> <ProcID> <InvokeID> 1$  
    <HeadsetID>‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### Data parameters

HeadsetID	Identifies the extension for the agent's headset. Up to 14 numeric characters, no special characters or embedded spaces.
-----------	--

#### Return values

Code	Description
E28850	Internal Proactive Contact system error: cannot open channel to operator monitor process. Unable to reserve headset ID. Might indicate an Proactive Contact system problem.
E28870	There is already a request to reserve the headset ID pending.
E28871	§ <HeadsetID> § <Message>àUnexpected return message from Proactive Contact. Message text follows headset ID.
E28922	No reserve headset ID request pending. Might indicate an Proactive Contact system problem.
E28923	§ <HeadsetID>àThe requested headset ID is reserved.
E50611	§ <HeadsetID>àThe requested headset ID is in use.
E50612	§ <HeadsetID>àNo more headsets are permitted on the system.
E50613	§ <HeadsetID>àFailure to access the headset ID table. Might indicate a permissions problem for the headset ID table file on the Proactive Contact system, a corrupt headset ID table file, or an Proactive Contact system problem.
M00000	Complete.
S28833	Pending.

## Examples

### Agent Application to Agent Binary

```
AGTReserveHeadset C COriginator_ID 11111 1 1$  
2‡
```

### Agent Binary to Agent Application

```
AGTReserveHeadset P Agent server 8497 1 2$  
0 $  
S28833‡  
AGTReserveHeadset R Agent server 8497 1 2$  
0 $  
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTConnHeadset](#)

[AGTFreeHeadset](#)

## AGTSendMessage

AGTSendMessage is a system command.

### Alternate name

SendMessage

### Function

Sends a message to the Proactive Contact supervisor process that displays on a supervisor screen.

Use this command to allow an agent to communicate with a supervisor.

### Availability

This command is available any time after AGTLogon.

### Format

```
AGTSendMessage C <Client> <ProclD> <InvokeID> 1$  
                  <Message>‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17



‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### Data parameters

Message	Text of the message to display on the supervisor's screen. Only one line on the screen is available for message display, so the maximum message length is 79 characters.
---------	--

#### Return values

Code	Description
M00000	Complete.

#### Examples

##### Agent Application to Agent Binary

```
AGTSendMessage C COriginator_ID 11111 1 1$
HELP ME‡
```

##### Agent Binary to Agent Application

```
AGTSendMessage R Agent server 5846 1 2$
0 $
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### See also

[AGTReceiveMessage](#)

## AGTSetCallback

AGTSetCallback is a working command.

#### Alternate name

SetCallback

## Function

Sets the recall time for a customer record. Proactive Contact automatically adjusts the recall time for the time zone.

Use this command to schedule customer recalls.

AGTListCallbackFmt gets the format of the date parameter and translates the format that the agent application uses into the Proactive Contact system time format.

It also lists the number of telephone numbers in the customer record.

The recall time format can be an absolute time or a time between the current time and the time when the recall should occur (an elapsed time).

If Proactive Contact configuration uses an elapsed time format, the vllocale.cfg file can define a minimum elapsed time.

## Availability

AGTSetCallback is available when an agent is working with a customer record. It is not necessary to have an open telephone line.

Proactive Contact cannot schedule recalls for inbound jobs.

## Format

AGTSetCallback can have from three to five parameters. It must include <Date>, <Time>, and <PhoneIndex>. Optional arguments include <RecallName> and <RecallNumber>. You can set the segments at five and leave blanks for the missing segments, or you can list only the relevant segments.

```
AGTSetCallback C <Client> <ProcID> <InvokeID> 5§  
                <Date> § <Time> § <PhoneIndex> § <RecallName> §  
                <RecallNumber> ‡
```

```
AGTSetCallback C <Client> <ProcID> <InvokeID> 3§  
                <Date> § <Time> § <PhoneIndex> ‡
```

```
AGTSetCallback C <Client> <ProcID> <InvokeID> 5§  
                <Date> § <Time> § <PhoneIndex> § <RecallName> §  
                <RecallNumber> ‡
```

```
AGTSetCallback C <Client> <ProcID> <InvokeID> 5§  
                <Date> § <Time> § <PhoneIndex> § §  
                <RecallNumber> ‡
```

```
AGTSetCallback C <Client> <ProcID> <InvokeID> 4§  
                <Date> § <Time> § <PhoneIndex> § <RecallName> ‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

‡ = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

Date	The date to place the recall. Eight or ten characters in one of the forms: YY/MM/DD, MM/DD/YY, or DD/MM/YY. *Must exist and match the format returned by AGTListCallbackFmt.
Time	The time of day when the recall should take place at the called party's location. Four or five characters in one of the forms: HHMM, HHMM[A or P], or HHMM+. The time represented is: 24-hour clock (HHMM) 12-hour clock (HHMM[A or P], where A=AM, P=PM) Incremental time (HHMM+) from current time
PhoneIndex	Index of the Proactive Contact phone field to use for the recall. Numeric, with a value usually between 1 and 4, where 1 corresponds to the PHONE1 field, 2 to the PHONE2 field, and so on. Must fall in the range returned by AGTListCallbackFmt.
RecallName	This is the customer name field to contact during the recall. There might be a default calling list field name to use for recalls specified in the RECALLNAME parameter of the job's fdict file. This argument is required when using RecallNumber. If a name does not exist, use an empty argument.
RecallNumber	This is the telephone number to use for the recall. There might be a default calling list field number to use for recalls specified in the RECALLNUMBER parameter of the job's fdict file. This parameter is optional.  This field is numeric, up to 43 characters, no embedded spaces or special characters. If the number is a complete telephone number rather than an extension, the format must match the standard format in the phone.cfg configuration file on the Proactive Contact system. When RecallNumber exits, Proactive Contact ignores the PhoneIndex entry.

\*Proactive Contact systems use the 10-character date format with 4-digit years, for example 1998/11/20.

**Return values**

Code	Description
------	-------------

Code	Description
E28800	§ <Minutes>àRecall cannot be less than <Minutes> from current time. This value is from the RECALLLIMIT settings in the vllocale.cfg file on the Proactive Contact system.
E28831	Date field has non-numeric value. Retry the command with the <Date> parameter set to a numeric value.
E28832	§<Input> § <DateFormat>‡ <Input> used as the date parameter does not match the Proactive Contact system <DateFormat> shown here. Retry command with correct date format.
E28833	Invalid month in date. Retry with month in the range 01 to 12.
E28834	Invalid year in date. Retry with year set to current year or current year plus some number.
E28835	Invalid day in date. Retry with a day value in the range for the current month. For example, use between 01 and 28 for February.
E28836	Invalid format character found. Check date for slash (/) versus backslash (\) and time for colons or other extra characters.
E28837	Invalid hour in time. If using the HHMM format, hour value must be between 01 and 24, for HHMM[A or P] hour value must be between 01 and 12.
E28838	Invalid minute in time. Minute value must fall in the range 00 to 59.
E28839	Invalid second in time. Retry command without any seconds value in the time parameter.
E28840	Time is not in correct format. Check for missing A, P, or + value following HHMM, missing digits in the hour or minute values or alphabetic values in the time.
E28841	Invalid phone index. The telephone field index does not fall in the range returned by AGTListCallbackFmt for the customer record. Retry with a value between 1 and the AGTListCallbackFmt value.
E28842	, <status> Invalid recall telephone selected. The telephone number specified by the <index> parameter is not valid. The <status> value is B, T, or Z. B indicates a bad telephone number, T indicates the phone number belongs to an unknown time zone, and Z indicates that the recall time is invalid in the time zone.

Code	Description
E28843	§ <PhoneNumber>àThe <PhoneNumber> is invalid. Either the format of the number does not match the standard phone format on Proactive Contact or the telephone number itself is not valid.
E28847	Date is before the current date. Retry the command with today's date or some future date.
E28848	Recall time and date outside time zone. The recall time setting is for the Proactive Contact system time. However, the telephone to recall is in a different time zone. In that time zone, the recall time setting is outside the legal boundaries for placing calls to customers.
E28849	, <Process> Because the time zone for the customer record is unknown, Proactive Contact cannot tell if the recall time the agent set is within legal calling times. <Process> discovered the bad time zone; it is always agent.
E28866	Telephone line is not available. There is a recall scheduled for the current date and time, but no telephone line is available to place the outbound call.
E28868	Recalls not permitted on inbound jobs. Proactive Contact cannot schedule recalls on inbound jobs.
M00000	Complete.

## Examples

### Agent Application to Agent Binary

```

AGTSetCallback C COriginator_ID 11111 152 5$
1998/03/01 $
13:05$
2$
Sue$
2107778888#
AGTSetCallback C COriginator_ID 11111 152 3$
98/03/02 $
06:00p$
1#
AGTSetCallback C COriginator_ID 11111 152 5$
98/02/28 $
01:00+$
$
Fred$
2107778888#
AGTSetCallback C COriginator_ID 11111 152 4$
98/02/28 $

```

```
01:00+$  
2$  
Tom‡
```

### Agent Binary to Agent Application

```
AGTSetCallback R Agent server 29722 152 2$  
0 $  
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTFinishedItem](#)

[AGTListCallbackFmt](#)

## AGTSetDataField

AGTSetDataField is a job setup command.

### Alternate name

SetDataField

### Function

Requests a calling list field to include with call notification or preview events. Each execution of AGTSetDataField adds a field to those sent with call notification and preview events.

Use this command to request customer information the agent needs when talking with the customer or deciding whether to call the customer.

AGTCallNotify and AGTPreviewRecord return the fields in the order the AGTSetDataField commands executed.

The field named in the command must match a field in the calling list for the attached job (see AGTListDataFields).

Fields set with AGTSetDataField remain in effect until either AGTClearDataSet or AGTDetachJob executes.

Proactive Contact does not use fields requested with this command to search for matching customer records.

AGTSetDataField does not affect the key field requested by AGTSetNotifyKeyField.

### Availability

This command is available any time after AGTAttachJob.

**Format**

AGTSetDataField C <Client> <ProclD> <InvokeID> 2§  
                   <ListType> § <FieldName>

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

ListType	Type of calling list used by the attached job: inbound (I) or outbound (O). Blend jobs use both I and O calling lists. Inbound jobs use type I calling lists. Outbound, Managed Dialing, Unit Work List, and Sales Verification jobs use type O calling lists.
FieldName	A field name from the calling list, as shown by AGTListDataFields. Up to 19 characters, alphanumeric, case sensitive, no embedded spaces.

**Return values**

Code	Description
E28885	Not attached to a job. Attach a job with AGTAttachJob and retry.
E28891	Invalid <ListType> parameter. Specify I (inbound) or O (outbound) calling list type.
E28892	No inbound calling list fields available. Retry using <ListType> O.
E28893	No outbound calling list fields available. Retry using <ListType> I.
E28894	The <FieldName> parameter does not match any of the field names in the calling list for the attached job. Execute AGTListDataFields to get correct field names.
M00000	Complete.

## Examples

### Agent Application to Agent Binary

```
AGTSetDataField C COriginator_ID 11111 178 2$  
  0 $  
  ACCTNUM‡
```

### Agent Binary to Agent Application

```
AGTSetDataField R Agent server 6111 178 2$  
  0 $  
  M00000‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTCallNotify](#)

[AGTClearDataSet](#)

[AGTDetachJob](#)

[AGTListDataFields](#)

[AGTPreviewRecord](#)

[AGTSetNotifyKeyField](#)

## AGTSetNotifyKeyField

AGTSetNotifyKeyField is a job setup command.

### Alternate name

SetNotifyKeyField

### Function

Requests the key field to include with the first message sent with call notification or preview events.

Use this command to set the search key for the matching customer record.

There can be only one key field. Each execution of AGTSetNotifyKeyField resets the key field for the attached job.

Executing AGTDetachJob automatically clears the key field setting.



**Availability**

Execute this command any time after executing AGTAttachJob. The field name included with the command must be a valid name from the attached job's calling list.

**Format**

```
AGTSetNotifyKeyField C <Client> <ProclD> <InvokeID> 2§
                        <ListType> § <FieldName> ‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

‡ = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

ListType	Type of calling list used by the attached job: inbound (I) or outbound (O). Blend jobs use both I and O calling lists. Inbound jobs use type I calling lists. Outbound, Managed Dialing, Unit Work List, and Sales Verification jobs use type O calling lists.
FieldName	A field name from the calling list, as shown by AGTListDataFields. Up to 19 characters, alphanumeric, case sensitive, no embedded spaces.

**Return values**

Code	Description
E28885	Not attached to a job. Execute AGTAttachJob and retry the command.
E28891	Must specify either inbound or outbound operation. Retry the command with the <ListType> parameter set to I or O.
E28892	No inbound calling list fields available. Retry with <ListType> O.
E28893	No outbound calling list fields available. Retry with <ListType> I.
E28894	<FieldName> not found. The <FieldName> parameter does not match any of the field names in the calling list used by the attached job. Execute AGTListDataFields to get correct field names.
M00000	Complete.

## Examples

### Agent Application to Agent Binary

```
AGTSendMessage C COriginator_ID 11111 1 1$  
HELP ME‡
```

### Agent Application to Agent Binary

```
AGTSetNotifyKeyField C COriginator_ID 11111 164 2$  
  O $  
  SYSNUM‡  
Agent Binary to Agent Application  
AGTSetNotifyKeyField R Agent server 15395 164 2$  
  O $  
  M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTAttachJob](#)

[AGTCallNotify](#)

[AGTDetachJob](#)

[AGTListDataFields](#)

## AGTSetPassword

AGTSetPassword is a system command.

### Alternate name

SetPassword

### Function

Sets the agent's password on Proactive Contact. Enables agents to change their password.

### Availability

This command is available when the agent is logged in to Proactive Contact.

### Format

AGTSetPassword <UserID> <PresentPW> <NewPW>

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### Data parameters

UserID	Agent's user ID.
PresentPW	Agent's current password.
NewPW	Agent's new password.

#### Password Rules

An Proactive Contact system password is case sensitive. It must contain 6 or more 7-bit ASCII characters from the USASCII character set with at least two alpha characters and one numerical or special character. The password cannot be the user's login name or any variation of the name. A new and current password must have at least three different characters.

#### Return values

Code	Description
E70013	Proactive Contact is not configured to allow an agent to change his or her password
E70014	Proactive Contact is configured with a locked password file. The agent cannot change his or her password.
E70015	The agent does not have access privileges to set a password. Contact your system administrator to reset a password.
E70016	The password the agent entered is not correct. Retry using a different password that follows the password rules.
E70017	The new password entry is incorrect. Retry using a different password that follows the password rules.

## AGTSetTenant

This command is used to set the tenant.

**Alternate name**

SetTenant

**Function**

Use this command to set tenant value that the agent selected. The selected tenant must be one of the values returned by AGTListTenants.

**Availability**

Execute this command after the agent login. If tenant name is provided with the agent name in AGTLogin, then there is no need to send this command.

**Format**

```
AGTSetTenant C <Client> <ProclD> <InvokeID> 1§
               <TenantName>‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

TenantName	Tenant name available for the agent. Use upto 20 charactors.
------------	--

**Return values**

Code	Description
E58018	Tenant is already set.
E58006	Failed to load tenant user mapping.
E58007	Unable to get tenant id for tenant name.
E58022	Tenant name is invalid or the use does not belong to the tenant.
M00000	Complete.

## Examples

### Agent Application to Agent Binary

```
AGTSetTenant C COriginator_ID 11111 172 1$  
Tenant1†
```

### Agent Binary to Agent Application

```
AGTSetTenant R Agent server 4839 172 2$  
0 $  
S28833‡  
AGTSetTenant R Agent server 4839 172 2$  
0 $  
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## AGTSetUnit

AGTSetUnit is a job setup command.

### Alternate name

SetUnit

### Function

Requests a Unit ID value for a Unit Work List job.

Use this command to request the Unit ID value the agent selected.

The selected Unit ID must be one of the values returned by AGTListUnits. Proactive Contact routes calls to an agent based on the Unit ID the agent selected at login.

If multi unit selection feature is enabled, agent can select multiple units.

“Unit Work List” dialog box of Moagent32.dll provides the facility of selecting multiple units. You can select multiple units in list of units. However, If you are not using “Unit Work List” dialog box of Moagent32.dll, then you have to provide the multiple units pipe (|) separated in AGTSetUnit API command.

Each execution of AGTSetUnit resets the Unit ID value.

AGTDetachJob clears the Unit ID value setting.

The following example illustrates how Proactive Contact uses Unit IDs.

1. The system supervisor sets the Unit ID field as LIMIT, the credit limit

2. AGTListUnits returns values of 2500, 5000, 7500, 10000. Each value is a Unit ID value for the job.
3. At login, the agent selects a Unit ID of 5000.
4. The agent application sends the AGTSetUnit command the value 5000.
5. Proactive Contact sends the agent only the calls for customers who have a credit limit of \$5000.

### Availability

Execute this command only with a Unit Work List job.

The agent must not yet be available for work.

The system does not carry the Unit ID value setting from job to job, even where jobs have the same units.

### Format

AGTSetUnit C <Client> <ProcID> <InvokeID> 1§  
                  <UnitID>‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### Data parameters

UnitID	<p>The Unit ID value selected by the agent. If you want to select multiple units then this data parameters contains pipe( ) separated units.</p> <p>Example: For single unit – U1 For multiple units - U1 U2 U3</p> <p><b>NOTE :</b> The interface of the AGTSetUnit API is not changed, it still accepts a data parameter, but in case of multiple units the units are pipe( ) separated.</p>
--------	--

### Return values

Code	Description
E28885	Not attached to a job; attach a Unit Work List job and retry.

Code	Description
E28886	The attached job is not a Unit Work List job. Detach the current job, attach a Unit Work List job, and retry.
E28887	Already available for work; cannot change Unit ID value. Execute the AGTNoFurtherWork command and retry.
E28888	The specified Unit ID value is not a Unit ID on the job. If necessary, use AGTListUnits command to list the units for the attached job. Select a valid Unit ID value and retry.
E29959	Number of selected units exceeded maximum allowed limit for a multi unit job. This Error Code is returned only if multi units are selected.
M00000	Complete.

## Examples

### Agent Application to Agent Binary

```
AGTSetUnit C COriginator_ID 11111 172 1$
230‡
```

### Agent Binary to Agent Application

```
AGTSetUnit R Agent server 4839 172 2$
0 $
M00000‡
```

### For multiple units:

### Agent Application to Agent Binary

```
AGTSetUnit C COriginator_ID 11111 172 1$
230|231|232|234‡
```

### Agent Binary to Agent Application

```
AGTSetUnit R Agent server 4839 172 2$
0 $
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**See also**

[AGTAttachJob](#)

[AGTDetachJob](#)

[AGTListUnits](#)

[AGTNoFurtherWork](#)

[AGTListUnits](#)

[AGTUnitEnd](#)

## AGTSetWorkClass

AGTSetWorkClass is a system command.

**Alternate name**

SetWorkClass

**Function**

Transmits the agent type to Proactive Contact.

Proactive Contact agent types are: Inbound, Outbound, Blend, Managed, and Person to Person.

If the agent application does not execute AGTSetWorkClass, the agent type defaults to outbound.

To route calls to an agent, the agent type must match the attached job type. For example, if the agent type is inbound, but the job is outbound, the AGTAvailWork command results in an error.

AGTSetWorkClass can be called anytime between the AGTLogon command and the AGTAvailWork.

The following table shows the relationship between agent types and job types.

Agent Type	Job Types
E28885	Not attached to a job; attach a Unit Work List job and retry.
Outbound	outbound, blend, Sales Verification, Unit Work List



Agent Type	Job Types
Blend1	blend
Managed	Managed Dialing
Person to Person	outbound, inbound, blend

To reset the agent type, execute `AGTSetWorkClass` with a different `<ClassID>`. The agent type setting carries from job to job until reset.

## Availability

Execute this command between the AGTLogon command and the AGTAvailWork command.

### Format

```
AGTSetWorkClass C <Client> <ProcID> <InvokeID> 1§
               <ClassID>±
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

± = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## Data parameters

ClassID	The code for the agent type. One character, alphabetic, case sensitive. Agent types codes are I (inbound), O (outbound), B (blend), P (Person to Person), and M (Managed),
---------	--

## Return values

Code	Description
E28882	The agent is available for work; cannot change class. Execute AGTNoFurtherWork and retry.
E28883	Invalid agent type. Retry with <ClassID> set to I, O, B, P, or M.

<sup>1</sup>Use the Blend agent type for Intelligent Call Blending Avaya Proactive Contact systems. For agents who handle Avaya Proactive Contact outbound and ACD inbound calls on Agent Blending Avaya Proactive Contact systems, use the AGTLogonAcd command to log in an Agent Blending agent.

Code	Description
M00000	Complete.

## Examples

### Agent Application to Agent Binary

```
AGTSetWorkClass C COriginator_ID 11111 168 1$
O‡
```

### Agent Binary to Agent Application

```
AGTSetWorkClass R Agent server 8497 168 2$
0 $
M00000‡
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## See also

[AGTAvailWork](#)

[AGTNoFurtherWork](#)

[AGTLogonAcd](#)

# AGTSTART

AGTSTART is a notification event message from the server.

This command name is case sensitive.

## Alternate name

START

## Function

Notifies the agent application that a connection with the agent binary on the Proactive Contact system has been established.

The client must respond with the AGTLogon command to start a session.

## Format

```
AGTSTART N <Agentserver> <ProcID> <InvokeID> 2$
0 $ AGENT_STARTUP‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII Data parameters

None.

#### Return values

Code	Description
E28858	,<AgentName> The <AgentName> logon exceeds the Proactive Contact system limit on the number of agents. This error can occur if client sessions for the agent application are left running on Proactive Contact after an agent workstation crashes.

#### Examples

##### Agent Binary to Agent Application

```
AGTSTART N Agent server 17970 0 2§  
0 §  
AGENT_STARTUP†
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### See also

[AGTLogon](#)

## AGTSystemError

AGTSystemError is a notification event message from the server.

#### Alternate name

SystemError

#### Function

Notifies the agent application that Proactive Contact has detected an internal error.

If the agent binary receives a message from Proactive Contact about the error, the agent binary terminates.

If the error is an I/O error between Proactive Contact and the agent binary, the agent binary continues. Consult the error message and Proactive Contact logs to determine the problem.

**Format**

AGTSystemError N <Agentserver> <ProclD> <InvokeID> 2§  
1 § <ErrorMessage>‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

ErrorMessage	The Proactive Contact error message number and any data or information that is part of that message.
--------------	--

**Return values**

Code	Description
E28859	, agent, <AgentName>, <slot_number> The agent logon is invalid. The first value is always agent, followed by the agent user name and the operator slot number Proactive Contact assigned to the agent name. Indicates an Proactive Contact system problem.
E28917	No job is attached. There is no current job that could have a job linked to it.
E28862	, <JobName> The agent's job selection (<JobName>) returned a fatal error. The agent child process supporting the client session is terminating.
E28863	Unknown file descriptor. Can indicate a command format problem from the agent application.
E28864	,<MessageText>Unknown IPC message. Message text follows.
E28921	Fatal error - terminating. Proactive Contact has experienced a critical internal error and is shutting down.
E28924	Must sign on system first. Must execute AGTLogon before any other command.

Code	Description
S28974	If for any reason an agent owned recall was aborted, the agent client application will receive the AGTSystemError notification with this code. It doesn't prevent the other AORs owned by that agent.
M00000	Complete.

## Examples

### Agent Binary to Agent Application

```
AGTSystemError N Agent server 4880 0 2$
  1 $
  E28924‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTDumpData](#)

[AGTLogloStart](#)

## AGTTransferCall

AGTTransferCall is a working command.

### Alternate name

TransferCall

### Function

Transfers a customer call on Proactive Contact systems configured for trunk-to-trunk transfers.

To transfer both the call and the customer record to another job, use AGTMoFlashBlind or AGTMoFlashSupv.

After Proactive Contact places the customer on hold, it uses a telephone line configured as a transfer trunk to place a manual call to another agent or supervisor.

After the transfer completes, the agent can speak with the person receiving the transfer call.

Then the agent application must execute one of the following commands:

- AGTTransferCall a second time with no <PhoneNumber> parameter to establish a conference call
- AGTFinishedItem to disconnect the agent's telephone connection and release the customer record
- AGTReleaseLine to disconnect the agent's telephone connection and continue to work with the customer record

## Availability

This command is available when the agent is working with a customer record and is talking with the customer.

This command is not available on an Avaya Proactive Contact with CTI system.

### Format

AGTTransferCall C <Client> <ProcID> <InvokeID> 1\$  
<PhoneNumber>‡

AGTTransferCall C <Client> <ProcID> <InvokeID> 1§  
‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

± = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

## Data parameters

PhoneNumber	The phone number to call. Numeric, up to 43 characters, no embedded spaces or special characters. If the number is a complete telephone number rather than an extension, the format must match the standard format in the phonefmt.cfg configuration file on the Proactive Contact system.
-------------	--

## Return values

Code	Description
E28628	Transfer failed - try again later.
E28866	Telephone line is not available. Proactive Contact cannot place the call. Retry later.
E28867	Telephone line is not off-hook. Retry the command with an active telephone connection to the customer.

Code	Description
E29950	This feature is not available on an Proactive Contact with CTI system.
M00000	Complete.

## Examples

### Agent Application to Agent Binary

```
AGTTransferCall C COriginator_ID 11111 122 1$
3425‡
```

### Agent Binary to Agent Application

```
AGTTransferCall R Agent server 2570 122 2$
0 $
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

[AGTHookflashLine](#)

[AGTManualCall](#)

[AGTMoFlashBlind](#)

[AGTMoFlashSupv](#)

## AGTUnholdCall

AGTUnholdCall is a working command.

### Alternate name

UnholdCall

### Function

Use this command to take a call off hold. It reestablishes a voice connection with a customer after issuing AGTHoldCall.

### Availability

This command is available when the agent is working with a customer record, is on the phone with the customer, and has previously placed the connection on hold.

This command is not available on an Avaya Proactive Contact with CTI system.

**Format**

AGTUnholdCall C <Client> <ProcID> <InvokeID> 0‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None.

**Return values**

Code	Description
E28866	Telephone line is not available. While there was a call on hold, it is not available to restore. The customer hung up while on hold.
E28867	Telephone line is not off-hook. There is no call on hold. Customer might have been cut off rather than placed on hold.
E29950	This feature is not available on an Proactive Contact with CTI system.
M00000	Complete.
S28814	Transfer is in progress. The call is in the process of being transferred.

**Examples****Agent Application to Agent Binary**

```
AGTUnholdCall C COriginator_ID 11111 135 0‡
```

**Agent Binary to Agent Application**

```
AGTUnholdCall R Agent server 2570 135 2§  
0 §  
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)



**See also**

[AGTHoldCall](#)

## AGTUnitEnd

AGTUnitEnd is a notification event message from the server.

**Alternate name**

UnitEnd

**Function**

This message is sent when the unit(s) the agent has logged into in a job have been exhausted by the dialer in a non-infinite job. In case the job is an infinite job, this event is not sent.

If the agent is working, AGTUnitEnd initiates AGTNoFurtherWork. The agent application must detach the job then log out or attach another job.

Introduced in Agent API version 5.0.0.0.4

**Format**

AGTUnitEnd N <Agentserver> <ProcID> <InvokeID> 2§  
0 § M00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None

**Return values**

Code	Description
M00000	Complete.

Examples

Agent Binary to Agent Application

```
AAGTUnitEnd N Agent server 17970 0 2$
0 $
M00000†
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)  
† = Message continues(ETB), ASCII x17

See also

- [AGTAttachJob](#)
- [AGTDetachJob](#)
- [AGTSetUnit](#)

AGTUpdateField

AGTUpdateField is a working command.

Alternate name

UpdateField

Function

Use this command to update the calling list during calling activities. The application might update any known field in either the outbound or inbound calling list.

Availability

This command is available when the agent is working with a customer record, either with or without an active telephone call to the customer.

Format

```
AGTUpdateField C <Client> <ProcID> <InvokeID> 3$
                <ListType> § <FieldName> § <NewValue>†
```

\$ = Message data separator, ASCII x1E (socket) or comma (tty)  
† = Message continues(ETB), ASCII x17

ListType	Type of calling list used by the attached job: inbound (I) or outbound (O). Blend jobs use both I and O calling lists. Inbound jobs use type I calling lists. Outbound, Managed Dialing, Unit Work List, and Sales Verification jobs use type O calling lists.
----------	--

FieldName	A field name from the calling list, as shown by AGTListDataFields. Up to 19 alphanumeric characters, case sensitive, no embedded spaces.
NewValue	A value to insert in the field. The value must fall within the parameters for the field returned by AGTListDataFields or AGTReadField, with regard to length and type of value (Alphanumeric, Numeric, Date, Time, or Currency).

**Return values**

Code	Description
E28885	Not attached to a job. Retry when attached to a job, available for work, and working with a customer record.
E28891	Must specify inbound or outbound operation. Retry with the <ListType> parameter set to either I or O.
E28892	No inbound calling list fields available. Retry with <ListType> O.
E28893	No outbound calling list fields available. Retry with <ListType> I.
E28894	<FieldName> not found. The system did not find the field name specified in the calling list. Field names are case sensitive and can contain underscores (_) in place of spaces.
E28912	Customer record is not available for update. The agent is not working with a customer record.
M00000	Complete.

**Examples****Agent Application to Agent Binary**

```
AGTSendMessage C COriginator_ID 11111 1 1$
HELP ME#
```

**Agent Application to Agent Binary**

```
AGTUpdateField C COriginator_ID 11111 179 3$
O $
PHONE2 $
2068821234#
```

### Agent Binary to Agent Application

```
AGTUpdateField R Agent server 2570 179 2$  
0 $  
M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### See also

[AGTListDataFields](#)

[AGTReadField](#)

## AGTXferCustHangup

AGTXferCustHangup is a notification event.

#### Alternate name

XferCustHangup

#### Function

Sent when a customer disconnects during a transfer. The event does not require turning on the AUTO RELEASE feature. Also, the phone line is not released automatically.

#### Availability

This command is available whenever trunk-to-trunk transfer calls are in progress.

#### Format

```
AGTXferCustHangup N <Agentserver> <ProcID> <InvokeID> 2$
```

```
0 § M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

#### Data parameters

None

**Return values**

Code	Description
M00000	Complete.

**See also**

[AGTAutoReleaseLine](#)

[AGTXferTrunkHangup](#)

## AGTXferTrunkHangup

AGTXferTrunkHangup is a notification event.

**Alternate name**

XferTrunkHangup

**Function**

Sent when a trunk disconnects during a transfer. The event does not require turning on the AUTO RELEASE feature. Also, the phone line is not released automatically.

**Availability**

This command is available whenever trunk-to-trunk transfer calls are in progress.

**Format**

AGTXferTrunkHangup N <Agentserver> <ProclD> <InvokeID> 2§

0 § M00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)

† = Message continues(ETB), ASCII x17

‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

**Data parameters**

None

**Return values**

Code	Description
------	-------------

Code	Description
M00000	Complete.

**See also**

[AGTAutoReleaseLine](#)

[AGTXferCustHangup](#)

## Appendix A: Completion codes

The following table shows the completion codes for Proactive Contact.

---

The values 20-34 and 51-85 are available for agent completion codes.

Code	Keyword	Type	Description	Report Header
000	NOTCALLED	System	The account has not been called.	
001	CODE1	System	Reserved for the system.	
002	ERROR	System	The system detected an invalid phone number.	
003	TIMEOUT	System	The system did not receive a dial tone.	Timed out
004	HANG_PORT	System	The line was idle after the system dialed the customer record phone number.	
005	NOTINZONE	System	The local time for the customer phone is outside calling hours.	Not within legal hours
006	MOFLASH_B	Agent	Used for native voice and data transfer. An agent transfers a call to an inbound agent without remaining on the line.	Blind transfer
007	HANG_TRANS	System	No agent is available for a supervisor transfer.	
008	TDSS_HF_B	Agent	ADAPTS API: the agent transfers a call without remaining on the call. This is known as blind hook flash transfer.	
009		System	Reserved for the system.	
010		System	Reserved for the system.	
011	BUSY	System	The system detected a busy signal.	

Code	Keyword	Type	Description	Report Header
012	CONTTONE	System	The system detected a continuous tone, such as a fax or modem.	
013	AUTOVOICE	System	The system detected an answering machine.	
014	VOICE	System	Interim code when a person is on the line.	
015	NOANSWER	System	The call placed was not answered.	
016	RINGING	Agent	Can be user defined but is usually defined as a phone call that was still ringing but was passed to an agent.	
017	CUSTHU	Agent	Can be user defined but is usually used to define when a customer hangs up while the call is in the wait queue, and the call is still passed to an agent.	
018	TRANSFER	Agent	Can be user defined but is usually defined as a transfer release.	Transferred
019	RECALL	Agent	Can be user defined but is usually defined as a recall release.	
020-034		Agent	Customer assigned codes used by agents.	
035	CANCEL	System	Can be user defined but is usually defined as the agent cancelled the managed call.	
036	INTERCEPT	System	Special Information Tone (SIT) received that indicates an operator intercepted the call.	
037	NOCIRCUIT	System	SIT received that indicates the circuits were unavailable.	



Code	Keyword	Type	Description	Report Header
038	DISCONN	System	SIT received that indicates the call was a disconnected number.	
039	VACANT	System	SIT received that indicates the call cannot be completed as dialed.	
040	REORDER	System	The call resulted in a fast busy tone.	
041	R_RINGING	System	Reserved.	
042	LINEFAIL	System	A failure on the phone line occurred.	
043	OP_RECALL	System	Operator set recall.	
044	DTMF_V	System	DTMF tone detected.	Voice DTMF
045	HU_INB	System	The customer hung up while in the inbound wait queue.	
046	HU_OUT	System	The customer hung up while in the outbound wait queue.	
047	HANG_INB	System	An agent was unavailable for the inbound call.	
048	HANG_OUT	System	An agent was unavailable for the outbound call.	
049	OPDIED	System	The agent session ended abnormally.	
050	R_HSONHOOK	System	The agent headset disconnected from Proactive Contact.	
051-088		Agent	Customer assigned codes used by agents.	
089	MANAGEDA	Agent	Managed Dial: Managed non-connection A.	

Code	Keyword	Type	Description	Report Header
090	MANAGEDDB	Agent	Managed Dial: Managed non-connection B.	
091	VIRTVOICE	System	Virtual Agent: Virtual message to VOICE, a person.	
092	VIRTAUTOV	System	Virtual Agent: Virtual message to AUTOVOICE, a calling machine.	
093	SOLD	Agent	Sales Verification: Sold campaign.	
094	VERIFIED	Agent	Sales Verification: Sale verified.	Verified sale
095	UNVERIFIED	Agent	Sales Verification: Sale not verified.	
096-097		System	Reserved for the system.	
098	AORECALL	Agent	Agent Owned Recall.	
099		System	Reserved for the system.	
100-200		Agent	Customer assigned	

## Appendix B: Agent API Message Format Quick Reference

The purpose of this section is to provide a quick reference to the command, response, data, and notification event records that make up the Proactive Contact Agent API (Agent API). Record formats appear in alphabetical order by type.

This section contains the following topics:

- [Command Messages](#)
- [Response Message Formats](#)
- [Data Message Formats](#)
- [Notification Event Message Formats](#)

### Command Message Formats

Command message formats in alphabetical order follow.

```
AGTAdjustHeadset C <Client> <ProcID> <InvokeID> 2 $<EarMouth> $<Volume>#
```

```
AGTAttachJob C <Client> <ProcID> <InvokeID> 1 $<JobName>#
```

```
AGTAvailWork C <Client> <ProcID> <InvokeID> 0#
```

```
AGTCancelSearch C <Client> <ProcID> <InvokeID> 0#
```

```
AGTClearDataSet C <Client> <ProcID> <InvokeID> 1 <ListType> #
```

```
AGTConnHeadset C <Client> <ProcID> <InvokeID> 0#
```

```
AGTDetachJob C <Client> <ProcID> <InvokeID> 0#
```

```
AGTDialDigit C <Client> <ProcID> <InvokeID> 1 $<Digit> #
```

```
AGTDisconnHeadset C <Client> <ProcID> <InvokeID> 0#
```

```
AGTDoNotCall C <Client> <ProcID> <InvokeID> 0#
```

```
AGTDumpData C <Client> <ProcID> <InvokeID> 1 $<FileName>#
```

```
AGTEchoOff C <Client> <ProcID> <InvokeID> 0#
```

```
AGTEchoOn C <Client> <ProcID> <InvokeID> 0#
```

```
AGTFindFirstRecord C <Client> <ProcID> <InvokeID> 1 $<KeyVal>#
```

```
AGTFindNextRecord C <Client> <ProcID> <InvokeID> 1 $<sNext>#
```

```
AGTFinishedItem C <Client> <ProcID> <InvokeID> 1 $<CompCode>#
```

```
AGTFreeHeadset C <Client> <ProcID> <InvokeID> 0#
```

```
AGTGetHeadsetVol C <Client> <ProcID> <InvokeID> 0#
```

## Avaya Proactive Contact Agent API 5.2

AGTHangupCall C <Client> <ProcID> <InvokeID> 0#

AGTHoldCall C <Client> <ProcID> <InvokeID> 0#

AGTHookflashLine C <Client> <ProcID> <InvokeID> 1 \$<PhoneNumber> #

AGTHookflashLine C <Client> <ProcID> <InvokeID> 1 \$#

AGTListCallbackFmt C <Client> <ProcID> <InvokeID> 0#

AGTListCallFields C <Client> <ProcID> <InvokeID> 1 \$<ListName>#

AGTListCallLists C <Client> <ProcID> <InvokeID> 0#

AGTListDataFields C <Client> <ProcID> <InvokeID> 1 \$<ListType> #

AGTListJobs C <Client> <ProcID> <InvokeID> 1 \$<JobType>#

AGTListKeys C <Client> <ProcID> <InvokeID> 0#

AGTListState C <Client> <ProcID> <InvokeID> 0#

AGTListUnits C <Client> <ProcID> <InvokeID> 0#

AGTLogIoStart C <Client> <ProcID> <InvokeID> 0#

AGTLogIoStop C <Client> <ProcID> <InvokeID> 0#

AGTLogoff C <Client> <ProcID> <InvokeID> 0#

AGTLogoffAcid C <Client> <ProcID> <InvokeID> 0#

AGTLogon C <Client> <ProcID> <InvokeID> 3 \$<AgentName> \$<Password>  
\$<AgentAPIVersion>#

AGTLogonAcid C <Client> <ProcID> <InvokeID> 2 \$<Extension> \$<PBX ID>#

AGTManagedCall C <Client> <ProcID> <InvokeID> 0#

AGTManualCall C <Client> <ProcID> <InvokeID> 1 \$<PhoneNumber>#

AGTMoFlashBlind C <Client> <ProcID> <InvokeID> 1 \$<JobName>#

AGTMoFlashSupv C <Client> <ProcID> <InvokeID> 1 \$<JobName>#

AGTMoFlashSupv C <Client> <ProcID> <InvokeID> 1 \$ #

AGTNoFurtherWork C <Client> <ProcID> <InvokeID> 0#

AGTReadField C <Client> <ProcID> <InvokeID> 2 \$<ListType> \$<FieldName>#

AGTReadyNextItem C <Client> <ProcID> <InvokeID> 0#

AGTReleaseLine C <Client> <ProcID> <InvokeID> 0#

AGTReleaseLine C <Client> <ProcID> <InvokeID> 1 \$<ScriptLabel>#

AGTReleaseLine C <Client> <ProcID> <InvokeID> 2 \$ \$<MessageNo>#

## Avaya Proactive Contact Agent API 5.2

```
AGTReserveHeadset C <Client> <ProcID> <InvokeID> 1 $<HeadsetID>#

AGTSendMessage C <Client> <ProcID> <InvokeID> 1 $<Message>#

AGTSetCallback C <Client> <ProcID> <InvokeID> 5 $<Date> $<Time> $<PhoneIndex> $
    <RecallName>$
    <RecallNumber>#

AGTSetCallback C <Client> <ProcID> <InvokeID> 3 $<Date>$<Time> $<PhoneIndex>#

AGTSetDataField C <Client> <ProcID> <InvokeID> 2 $<ListType> $<FieldName> #

AGTSetNotifyKeyField C <Client> <ProcID> <InvokeID> 2 $<ListType> $<FieldName>#

AGTSetPassword C <Client> <ProcID> <InvokeID> 3 $<UID> $<Old> $<New>#

AGTSetUnit C <Client> <ProcID> <InvokeID> 1 $<UnitID>#

AGTSetWorkClass C <Client> <ProcID> <InvokeID> 1 $<ClassID>#

AGTTransferCall C <Client> <ProcID> <InvokeID> 1 $<PhoneNumber>#

AGTTransferCall C <Client> <ProcID> <InvokeID> 1 $ #

AGTUnholdCall C <Client> <ProcID> <InvokeID> 0#

AGTUpdateField C <Client> <ProcID> <InvokeID> 3 $<ListType> $<FieldName>
$<NewValue>#
```

## Response Message Formats

All commands receive the generic complete message (Proactive Contact code M00000) when the action is complete. Many commands receive the generic pending message (Proactive Contact code S28833) that indicates a command requires processing by Proactive Contact.

```
<Generic>RAgent server<ProcID><InvokeID>2$ 0 $ M00000#

<Generic>PAgent server<ProcID><InvokeID>2$ 0 $ S28833#
```

## Data Message Formats

```
AGTGetHeadsetVol D Agent <ProcID> <InvokeID> 4$0$
server M00001$
<Ear>$
Mouth>#

AGTListCallbackFmt D Agent <ProcID> <InvokeID> 4 $0$
server M00001$
<Format>$
<Phones>#

AGTListCallFields D Agent <ProcID> <InvokeID> <n> $0$
server M00001$
<FieldName>,
<FieldLength>,
```

## Avaya Proactive Contact Agent API 5.2

```
<FieldType>, F $  
<FieldName>,  
<FieldLength>,  
<FieldType>, F ...#
```

--- See below ---

The <FieldType> is C (character), N (numeric), \$(currency), D (date), or T (time).

The F value is a placeholder for future use.

```
AGTListCallLists D Agent <ProcID> <InvokeID> <n> $ 0 $  
server M00001$  
<ListName1> $  
<ListName2>...#
```

```
AGTListDataFields D Agent <ProcID> <InvokeID> <n> $ 0 $  
server M00001 $  
<FieldName>,  
<FieldType>, F $  
<FieldName>,  
<FieldLength>,  
<FieldType>, F ...#
```

```
AGTListKeys D Agent <ProcID> <InvokeID> <n> $ 0 $ M00001 $  
server <CompCode>,  
<Description>,  
<ScriptLabel> $  
<CompCode>,  
<Description>,  
<ScriptLabel> $...#
```

```
AGTListJobs D Agent <ProcID> <InvokeID> <n> $ 0 $ M00001 $  
server <JobType>,  
<JobName>, <Status> $  
<JobType>,  
<JobName>, <Status>. .  
. #
```

```
AGTListState D Agent <ProcID> <InvokeID> 2 $0$  
server S70000,<JobName>#
```

```
AGTListState D Agent <ProcID><InvokeID> 2 $0$  
server S70001,<JobName>#
```

```
AGTListState D Agent <ProcID> <InvokeID> 2 $0$  
server S70002,<JobName>#
```

```
AGTListState D Agent <ProcID> <InvokeID> 2 $0$  
server S70003,<JobName>#
```

```
AGTListState D Agent <ProcID> <InvokeID> 2 $0$  
server S70004#
```

```
AGTListUnits D Agent <ProcID> <InvokeID> <n> $0$  
server M00001$  
<UnitID>,  
<UnitID>...#
```

## Avaya Proactive Contact Agent API 5.2

```
AGTManagedCall D Agent <ProcID> <InvokeID> 3 $0$  
server M00001$  
<CallStatus>#
```

```
AGTManualPhone D Agent <ProcID> <InvokeID> 3 $ 0 $  
M00001 $ <Formatted PhoneNumber>#
```

```
AGTReadField D Agent <ProcID> <InvokeID> 3 $0$  
server M00001$  
<FieldName>,  
<FieldType>,  
<FieldLength>,  
<FieldValue> #
```

```
AGTFindFirstRecord D Agent <ProcID> <InvokeID> <n> $0$  
server M00001$  
<Message No.>,<Records Found>,<Current Record>#
```

```
AGTFindFirstRecord D Agent <ProcID> <InvokeID> <n> $0$  
server M00001$  
<Records Found> records found, <Current Record> is being displayed$  
MANAGED$  
<Notify Key Field Name>, <Value>#
```

```
AGTFindFirstRecord D Agent <ProcID> <InvokeID> <n> $0$  
server M00001$  
<Field Name>, <Value>#
```

```
AGTFindNextRecord D Agent <ProcID> <InvokeID> <n> $0$  
server M00001$  
<Message No.>,<Records Found>,<Current Record>#
```

```
AGTFindNextRecord D Agent <ProcID> <InvokeID> <n> $0$  
server M00001$  
<Records Found> records found, <Current Record> is being displayed$  
MANAGED$  
<Notify Key Field Name>, <Value>#
```

```
AGTFindNextRecord D Agent <ProcID> <InvokeID> <n> $0$  
server M00001$  
<Field Name>, <Value>#
```

## Notification Event Message Formats

```
AGTAutoReleaseLine N <Agentserver> <ProcID> <InvokeID> 2$  
0 $ M00000#
```

```
AGTCallNotify N Agent <ProcID> <InvokeID> 5 0  
server $ M00001  
$<OpMesg>[*<WaitMsg>]  
$<CallType>  
$<NotifyFieldName>,  
<NotifyFieldData> #
```

```
AGTCallNotify N Agent <ProcID> <InvokeID> <n> $ 0  
server $ M00001
```

## Avaya Proactive Contact Agent API 5.2

```
$ <FieldName>,  
<FieldData>  
$<FieldName>,  
<FieldData>...+  
  
AGTHeadsetConnBrokenN Agent <ProcID> <InvokeID> 2 $1$  
server E28800+  
  
AGTIicbAbort N Agent <ProcID> <InvokeID> 2 $0$  
server M00000+  
  
AGTIicbFeNotif N Agent <ProcID> <InvokeID> 2 $0$  
server S28971+  
  
AGTIicbFeNotif N Agent <ProcID> <InvokeID> 2 $0$  
server S28972+  
  
AGTIicbOffline N Agent <ProcID> <InvokeID> 2 $0$  
server M00000+  
  
AGTIicbOnline N Agent <ProcID> <InvokeID> 2 $0$  
server M00000+  
  
AGTJobEnd N Agent <ProcID> <InvokeID> 2 $0$  
server M00000+  
  
AGTJobTransLink N Agent <ProcID> <InvokeID> 3 $0$  
server M00000 $  
<JobName> +  
  
AGTJobTransRequest N Agent <ProcID> <InvokeID> 3 $0$  
server M00000 $  
<JobName>+  
  
AGTManCallAnswered N Agent <ProcID> <InvokeID> 2$  
0 $ M00000+  
  
AGTPreviewRecord N Agent <ProcID> <InvokeID> 6 $0$  
server M00001$  
<OpMesg>$  
<CallType>$  
<NotifyFieldName>,$  
<FieldData>+  
  
AGTPreviewRecord N Agent <ProcID> <InvokeID> 3 $ 0$  
server M00001$  
<FieldName>,<FieldData>$  
<FieldName>,<FieldData>  
...+  
  
AGTReceiveMessage N Agent 1268 0 2 $ 0 $  
server M00000+  
  
AGTSTART N Agent <ProcID> <InvokeID> 2 $0$  
server AGENT_STARTUP+  
  
AGTSystemError N Agent <ProcID> <InvokeID> 2 $1$  
server <ErrorMessage>à$
```



## Avaya Proactive Contact Agent API 5.2

AGTXferCustHangup N <Agentserver> <ProcID> <InvokeID> 2\$  
0 \$ M00000‡

AGTXferTrunkHangup N <Agentserver> <ProcID> <InvokeID> 2\$  
0 \$ M00000‡

AGTJobMode N <Agentserver> <ProcID> <InvokeID> 4\$ 0\$  
S28996 \$ 1 \$ 1‡

AGTJobMode N <Agentserver> <ProcID> <InvokeID> 2\$ 0\$  
S28997‡

## Appendix C: System Messages

The purpose of this section is to provide a table of system messages that might be returned to the Avaya Agent API (Agent API).

This section contains the following topics:

- [ErrorMessages](#)
- [Data Messages](#)
- [Pending Messages](#)

### Error Messages

In the information that follows, each message number has its own heading. For each message number, the tables contain the following information:

- The Agent API commands that might return the message
- A message interpretation for each Agent API command
- The text from the Moagent32.ini file for the message

Some error messages indicate an Proactive Contact system problem. For assistance with these errors, contact the Proactive Contact system administrator.

---

Proactive Contact might occasionally return error messages to the Agent API that do not appear in this listing. For assistance with these error messages, please contact your Avaya representative.

#### E12152

Source	Message Interpretation
AGTFindNextRecord	This error will come when the previous record is not present.
ErrorText	No Previous Record

#### E12153

Source	Message Interpretation
AGTFindNextRecord	End of record is reached. There is no next record
ErrorText	No More Record

#### E12156

Source	Message Interpretation
--------	------------------------

Source	Message Interpretation
AGTFindFirstRecord	There are not records present in list that matches the search key value
ErrorText	No Record Found

**E00518,agent, <ListName>**

Source	Message Interpretation
AGTListCallFields	There is no calling list with <ListName> on the system. The agent entry is the name of the process on the Proactive Contact system returning the error. <ListName> is the name submitted by the client process.
AGTListKeys	Unable to open the keys file. Indicates an Proactive Contact system problem.
Error text	E00518: Calling list %s does not exist.

**E28628**

Source	Message Interpretation
AGTTransferCall AGTMoFlashSupv	Transfer is failed.
Error text	Transfer failed - try again later.

**E28800 § <minutes>†**

Source	Message Interpretation
AGTSetCallback	Recall cannot be less than <minutes> from current time. This value is from the RECALLLIMIT settings in the vlocale.cfg file on the Proactive Contact system.
Error text	E28800: Recall cannot be less than %s minutes.

**E28804**

Source	Message Interpretation
AGTAttachJob	Job <JobName> is not running, not active.
Error text	E28804: Job %s is not running.

**E28805 § <JobName> ‡**

Source	Message Interpretation
AGTAvailWork	<JobName> is not ready. The attached job is active, but is not placing calls. Retry the login after job starts placing calls.
Error text	E28805: Job %s is not ready. Join the job later.

**E28812**

Source	Message Interpretation
AGTLogon	<AgentName> is logged in to another application process. The agent might be logged in at another location. Retry the login with a different <AgentName>.
Error text	E28812: Agent %s already logged on. Access is denied.

**E28813**

Source	Message Interpretation
AGTAvailWork	No more agents of this type can join the job. Job parameters include a maximum number of agents of each type (work class) that might log into the job. See <a href="#">AGTSetWorkClass</a> on page 208.
Error text	E28813: Maximum %s agent limit reached.

**E28814**

Source	Message Interpretation
--------	------------------------

Source	Message Interpretation
AGTAvailWork	Managed agents might not join this job. The agent application agent type (work class) is M (Managed). The attached job is not a Managed Dialing job. Reset the agent type and retry.
Error text	E28814: Managed agents cannot join this job.

**E28815**

Source	Message Interpretation
AGTAvailWork	The agent is logged in as a Unit Work List agent and is attempting to join a Sales Verification job. Select a different job and retry.
Error text	E28815: Sales verification with unit work lists is not permitted.

**E28816**

Source	Message Interpretation
AGTAvailWork	Only inbound agents can join this job. The agent application agent type (work class) does not match the job type. Set the agent type to I and retry.
Error text	E28816: Only inbound agents are permitted.

**E28817**

Source	Message Interpretation
AGTAvailWork	Only outbound agents can join this job. The agent application agent type (work class) does not match the job type. Set the agent type to O and retry.
Error text	E28817: Only outbound agents are permitted.

**E28818**

Source	Message Interpretation
--------	------------------------

Source	Message Interpretation
AGTAvailWork	Only Outbound or Managed agents can join this job. The agent application agent type (work class) does not match the job type. Set the agent type to O or M and retry.
Error text	E28818: Only outbound or Managed agents are permitted.

**E28819**

Source	Message Interpretation
AGTAvailWork	Only Outbound agents can join Sales Verification jobs. The agent application agent type (work class) does not match the job type. Set the agent type to O and retry.
Error text	E28819: Only outbound agents are permitted on a Sales Verification job.

**E28831**

Source	Message Interpretation
AGTSetCallback	Date field has non-numeric value. Retry the command with the <Date> parameter set to a numeric value.
Error text	E28831: Field has non-numeric value.

**E28832 § <input>§ <DateFormat> ‡**

Source	Message Interpretation
AGTSetCallback	<input> used as the date parameter does not match the Proactive Contact <DateFormat> shown here. Retry command with correct date format.
Error text	E28832: Date ccyy/mm/dd %s submitted doesn't match %s format.

**E28833**

Source	Message Interpretation
--------	------------------------

Source	Message Interpretation
AGTSetCallback	Invalid month in date. Retry with month in the range 01 to 12.
Error text	E28833: Date has an invalid month.

**E28834**

Source	Message Interpretation
AGTSetCallback	Invalid year in date. Retry with year set to current year or current year plus some number.
Error text	E28834: Date has an invalid year.

**E28835**

Source	Message Interpretation
AGTSetCallback	Invalid day in date. Retry with a day value in the range for the current month. For example, use between 01 and 28 for February.
Error text	E28835: Date has an invalid day.

**E28836**

Source	Message Interpretation
AGTSetCallback	Invalid format character found. Check date for slash (/) versus backslash (\) and time for colons or other extra characters.
Error text	E28836: Invalid format character found.

**E28837**

Source	Message Interpretation
AGTSetCallback	Invalid hour in time. If using the HHMM format, hour value must be between 01 and 24; for HHMM[A or P] hour value must be between 01 and 12.

Source	Message Interpretation
Error text	E28837: Time has an invalid hour.

**E28838**

Source	Message Interpretation
AGTSetCallback	Invalid minute in time. Minute value must fall in the range 00 to 59.
Error text	E28838: Time has an invalid minute.

**E28839**

Source	Message Interpretation
AGTSetCallback	Invalid second in time. Retry command without any seconds value in the time parameter.
Error text	E28839: Time has an invalid second.

**E28840**

Source	Message Interpretation
AGTSetCallback	Time is not in correct format. Check for missing A, P, or + value following HHMM, missing digits in the hour or minute values, or alphabetic values in the time.
Error text	E28840: Time is not in the correct format.

**E28841**

Source	Message Interpretation
AGTSetCallback	Invalid phone index. The telephone field index does not fall in the range returned by AGTListCallbackFmt for the customer record. Retry with a value between 1 and the AGTListCallbackFmt value.
Error text	E28841: Invalid phone.



**E28842, <status>**

Source	Message Interpretation
AGTSetCallback	Invalid recall telephone selected. The telephone number specified by the <index> parameter is not valid. The <status> value is B, T, or Z. B indicates a bad telephone number, T indicates the phone number belongs to an unknown time zone, and Z indicates that the recall time is invalid in the time zone.
Error text	E28842: Invalid phone number. Phonestat: %s.

**E28843, <PhoneNumber>**

Source	Message Interpretation
AGTManualCall	The <PhoneNumber> is invalid. Either the format of the number does not match the standard phone format on Proactive Contact, or the telephone number itself is not valid.
Error text	E28843: Invalid phone number - %s.

**E28847**

Source	Message Interpretation
AGTSetCallback	Date is before the current date. Retry the command with today's date or some future date.
Error text	E28847: Date is before the current date.

**E28848**

Source	Message Interpretation
AGTSetCallback	Recall time and date outside time zone. The recall time setting is for the Proactive Contact system time. However, the telephone to recall is in a different time zone. In that time zone, the recall time setting is outside the legal boundaries for placing calls to customers.
Error text	E28848: Recall time is outside the limits for the time zone.

**E28849, <process>**

Source	Message Interpretation
AGTSetCallback	Because the time zone for the customer record is unknown, Proactive Contact cannot tell if the recall time is within legal calling times. <Process> discovered the bad time zone; it is always agent.
Error text	E28849: %s - Time zone for the record is not known.

**E28850**

Source	Message Interpretation
AGTConnHeadset	Internal Proactive Contact system error: cannot open a channel to the operator monitor process; unable to connect headset ID. This message might indicate an Proactive Contact system problem.
AGTReserveHeadset	Internal Proactive Contact system error: cannot open channel to the operator monitor process; unable to reserve headset ID. Might indicate an Proactive Contact system problem.
AGTAdjustHeadset AGTGetHeadsetVol	Cannot open the channel to the operator monitor process. Indicates an Proactive Contact system problem.
Error text	E28850: ERROR: Cannot open channel to operator monitor process.

**E28851**

Source	Message Interpretation
AGTGetHeadsetVol	There is no response from the operator monitor process. Indicates an Proactive Contact system problem.
Error text	E28851: No response from the operator monitor process.

**E28858, <AgentName>**

Source	Message Interpretation
--------	------------------------

Source	Message Interpretation
AGTSTART	The <AgentName> login exceeds the Proactive Contact limit on the number of agents. This error can occur if client sessions for the agent application are left running on the Proactive Contact system after agent workstation crashes.
Error text	E28858: %s exceeded the number of agent slots available.

**E28859, <agent>, <AgentName>, <slot\_number>**

Source	Message Interpretation
AGTSystemError	The agent login is invalid. The first value is always agent, followed by the agent login name and the operator slot number Proactive Contact assigned to the agent name. Indicates an Proactive Contact system problem.
Error text	E28859: %s (%s) - Agent number returned invalid %s.

**E28862, <JobName>**

Source	Message Interpretation
AGTSystemError	The agent's job selection (<JobName>) returned a fatal error. The agent child process supporting the client session is terminating.
Error text	E28862: Proactive Contact FATAL ERROR ON SELECT %s. Process terminating.

**E28863**

Source	Message Interpretation
AGTSystemError	Unknown file descriptor. Can indicate a command format problem from the agent application.
Error text	E28863: ERROR. Unknown file descriptor.

**E28864, <MessageText>**

Source	Message Interpretation
AGTSystemError	Unknown IPC message. Message text follows.
Error text	E28864: Unknown IPC message - %s.

**E28865, <Command>**

Source	Message Interpretation
Generic	Unknown command message. The <Command> is the text string that agent received as the command.
Error text	E28865: Unknown Command message - %s.

**E28866**

Source	Message Interpretation
AGTAdjustHeadset AGTDialDigit AGTGetHeadsetVol AGTManualCall	Telephone line is not available. The agent must receive a customer call to acquire an open telephone line.
AGTFinishedItem	There is no line available. This error generally indicates the agent is trying to cancel a Managed Dialing call too late.
AGTHangupCall	A telephone line is not available. There is no call to hang up.
AGTHoldCall	Telephone line is not available. There is no open telephone line to place on hold.
AGTHookflashLine AGTMoFlashBlind AGTMoFlashSupv AGTTransferCall	No phone line is available. Proactive Contact does not have a telephone line available to place the transfer call.
AGTReleaseLine	Failed to release the line for the Managed Dialing preview (no telephone line available). This message results when Proactive Contact is placing the preview call.

Source	Message Interpretation
AGTSetCallback	Telephone line is not available. There is a recall scheduled for current date and time, but no telephone line is available to place the outbound call.
AGTUnholdCall	Telephone line is not available. While there was a call on hold, it is not available to restore. The customer hung up while on hold.
Error text	E28866: Telephone line is not available.

**E28867**

Source	Message Interpretation
AGTAdjustHeadset	Telephone line is not offhook. The agent has an open telephone line that is on hold.
AGTDialDigit AGTGetHeadsetVol	The telephone line is not offhook. Place the telephone line in offhook state and retry.
AGTHoldCall	Telephone line is not offhook. While there is an open telephone line, there is no call to place on hold. This message might mean the agent has already placed the call on hold.
AGTMoFlashBlind AGTMoFlashSupv AGTHookflashLine AGTTransferCall	Line is not offhook. There is no current telephone call to transfer.
AGTUnholdCall	Telephone line is not offhook. There is no call on hold. Customer might have been cut off rather than placed on hold.
Error text	E28867: Telephone line is not offhook.

**E28868**

Source	Message Interpretation
AGTListCallbackFmt	No recalls during inbound jobs. The current attached job is an inbound job. There is no recall format.

Source	Message Interpretation
AGTMoFlashSupv	The transfer failed, try again later. There is no agent available on the specified job to take the transfer call.
AGTSetCallback	Recalls not permitted on inbound jobs. Proactive Contact cannot schedule recalls on inbound jobs.
Error text	E28868: Recalls are not permitted on inbound calls.

**E28869**

Source	Message Interpretation
AGTAdjustHeadset	Headset volume must be in the range of one to eight. Retry the command using a number from one to eight in the <Volume> parameter.
Error text	E28869: Headset volume must be set between 1 and 8.

**E28870**

Source	Message Interpretation
AGTReserveHeadset	There is already a request to reserve the headset ID pending.
Error text	E28870: Reserve headset ID request pending.

**E28871 § <HeadsetID>§ <Message>‡**

Source	Message Interpretation
AGTReserveHeadset	Unexpected return message from Proactive Contact. Message text follows headset ID.
Error text	E28871: Invalid headset ID - %s.

**E28872**

Source	Message Interpretation
--------	------------------------

Source	Message Interpretation
AGTConnHeadset	Headset is already connected.
AGTLogonAcd	The agent is already logged in to ACD.
Error text	E28872: Headset is already connected.

**E28873**

Source	Message Interpretation
AGTConnHeadset	Headset ID is not reserved. Execute AGTReserveHeadset and retry.
AGTDisconnHeadset	The headset is not reserved. There is no headset to disconnect.
AGTFreeHeadset	There is no headset reserved. There is nothing to free.
AGTLogonAcd	Headset is not reserved. Reserve a headset identification with AGTReserveHeadset before executing this command.
Error text	E28873: Headset ID is not reserved nor validated.

**E28874**

Source	Message Interpretation
AGTConnHeadset	A connect headset request is already pending.
Error text	E28874: Connect headset request is pending.

**E28875**

Source	Message Interpretation
AGTConnHeadset	The headset connect request did not register. This message might indicate an Proactive Contact system problem.
Error text	E28875: No headset connect request is pending.

**E28876**

<b>Source</b>	<b>Message Interpretation</b>
AGTConnHeadset	The headset is not connected. This message might indicate that there is a problem with the telephone connection to the headset or an Proactive Contact system problem.
AGTDisconnHeadset	The headset is not connected. There is no headset connection to close.
Error text	E28876: Headset is not connected.

**E28877**

<b>Source</b>	<b>Message Interpretation</b>
AGTDisconnHeadset	There is a disconnect headset request already pending.
Error text	E28877: Disconnect headset request is pending.

**E28879**

<b>Source</b>	<b>Message Interpretation</b>
AGTFreeHeadset	The headset is not disconnected. Disconnect the headset and retry.
Error text	E28879: Headset is not disconnected.

**E28880**

<b>Source</b>	<b>Message Interpretation</b>
AGTHeadsetConnBroken	Headset connection is broken.
Error text	E28880: Headset connection is broken.

**E28881**

<b>Source</b>	<b>Message Interpretation</b>
---------------	-------------------------------



Source	Message Interpretation
AGTHeadsetConnBroken	Headset connection re-established.
Error text	E28881: Headset reconnected.

**E28882**

Source	Message Interpretation
AGTSetWorkClass	The agent is available for work; cannot change class. Execute AGTNoFurtherWork and retry.
Error text	E28882: Already available for work. Cannot change the agent type (work class).

**E28883**

Source	Message Interpretation
AGTSetWorkClass	Invalid agent type. Retry with <ClassID> set to I, O, B, P, or M.
Error text	E28883: Invalid agent type (work class). Type must be B, I, O, M, or P.

**E28884**

Source	Message Interpretation
AGTListJobs	The agent server was unable to access Proactive Contact shared memory; usually indicates an Proactive Contact system problem.
Error text	E28884: Cannot attach to shared data memory.

**E28885**

Source	Message Interpretation
AGTAvailWork	The agent application is not attached to a job. The agent must select a job before joining it.

Source	Message Interpretation
AGTCallNotify AGTPreviewRecord	The agent application is not attached to a job. The event is inappropriate.
AGTClearDataSet AGTFinishedItem AGTListCallbackFmt AGTListDataFields AGTListKeys AGTSetDataField AGTSetNotifyKeyField	The agent application is not attached to a job. Execute AGTAttachJob and retry.
AGTListUnits AGTSetUnit	Not attached to a job; attach an active Unit Work List job and retry.
AGTFinishedItem AGTManagedCall AGTReadField AGTReleasedLine AGTUpdateField	Not attached to a job. Retry when attached to a job, available for work, and working with a customer record.
AGTReadyNextItem	Not attached to a job. Attach a job, become available for work, and retry.
Error text	E28885: Not attached to a job.

**E28886**

Source	Message Interpretation
AGTListUnits AGTSetUnit	The attached job is not a Unit Work List job; change attached job and retry.
Error text	E28886: Unit work lists are not permitted on this job.

**E28887**

Source	Message Interpretation
--------	------------------------

Source	Message Interpretation
AGTSetUnit	Already available for work; cannot change unit ID value. Execute the AGTNoFurtherWork command and retry.
Error text	E28887: Already available for work. Cannot change unit.

**E28888**

Source	Message Interpretation
AGTSetUnit	The specified unit ID value is not a unit ID on the job; retry. If necessary use AGTListUnits command to list the units for the attached job. Select a valid unit ID value and retry.
Error text	E28888: Unit not found.

**E28889**

Source	Message Interpretation
AGTAttachJob	The agent application is already attached to a job. The agent must clear the current job selection before selecting a new one.
Error text	E28889: Already attached to a job. Detach current job and retry.

**E28890**

Source	Message Interpretation
AGTAttachJob	Cannot open the job's resource file. This message usually indicates an Proactive Contact system problem.
Error text	E28890: Failure to open job % resource file.

**E28891**

Source	Message Interpretation
--------	------------------------

Source	Message Interpretation
AGTClearDataSet	Must specify inbound or outbound operation. Retry the command with the <ListType> parameter set to either I or O.
AGTListDataFields AGTReadField AGTSetDataField AGTSetNotifyKeyField AAGTUpdateField	Must specify I (inbound) or O (outbound) calling list type. Retry command with <ListType> parameter included.
Error text	E28891: Must specify INBOUND or OUTBOUND operation.

**E28892**

Source	Message Interpretation
AGTClearDataSet AGTListDataFields AGTReadField AGTSetDataField AGTSetNotifyKeyField AGTUpdateField	There are no inbound calling list fields available. Retry with <ListType> O.
Error text	E28892: No inbound calling list fields are available.

**E28893**

Source	Message Interpretation
AGTClearDataSet AGTListDataFields AGTReadField AGTSetDataField AGTSetNotifyKeyField AGTUpdateField	No outbound calling list fields available. Retry with <ListType> I.
Error text	E28893: No outbound calling list fields are available.

**E28894**

Source	Message Interpretation
AGTSetNotifyKeyField AGTSetDataField	<FieldName> not found. The <FieldName> parameter does not match any of the field names in the calling list used by the attached job. Execute AGTListDataFields to get correct field names.
AGTReadField	<FieldName>not found. Retry with valid field name for the calling list.
AGTUpdateField	<FieldName> not found. The system did not find the field name specified in the calling list. Field names are case sensitive and might contain underscores (_) in place of spaces.
Error text	E28894: <FieldName> not found.

**E28895**

Source	Message Interpretation
AGTAvailWork	The agent is already available for work.
Error text	E28895: Already available for work.

**E28896**

Source	Message Interpretation
AGTAvailWork	The agent's headset must be active. Execute AGTConnHeadset and retry.
AGTReadyNextItem	Headset must be active. The agent's headset is disconnected. Reconnect the headset and retry.
Error text	E28896: Headset must be active.

**E28897**

Source	Message Interpretation
AGTAvailWork	An available for work request is already pending.

Source	Message Interpretation
Error text	E28897: Available for work request is pending.

**E28898**

Source	Message Interpretation
AGTAvailWork	The job is not available for login. The job might have become inactive since it was attached. Execute AGTListJobs to check the job status.
Error text	E28898: Job is not available.

**E28899**

Source	Message Interpretation
AGTAvailWork	There is no available for work request pending. AGTAvailWork did not execute. This message might indicate an Proactive Contact system problem.
Error text	E28899: No available for work request is pending.

**E28900 § <message>‡**

Source	Message Interpretation
AGTAvailWork AGTCallNotify AGTDisconnHeadset AGTHeadsetConnBroken AGTJobTransLink AGTJobTransRequest AGTPreviewRecord AGTReceiveMessage AGTManagedCall	An Proactive Contact system internal error occurred. Error message text follows.
Error text	E28900: Wrong message ID received - %s.

**E28901**

Source	Message Interpretation
AGTJobTransRequest	Agent is not available for work. The AGTNoFurtherWork command initiated by agent is not necessary.
AGTManagedCall AGTReadyNextItem	Agent is not available for work. Retry after executing AGTAvailWork.
Error text	E28901: Not available for work.

**E28902**

Source	Message Interpretation
AGTReadyNextItem	Already on a customer record. Release the current record with AGTReleaseLine or AGTFinishedItem and retry.
Error text	E28902: Already have open customer record.

**E28903**

Source	Message Interpretation
AGTReadyNextItem	Already set ready for next customer record. AGTReadyNextItem already executed.
Error text	E28903: Already set ready for next customer record.

**E28904**

Source	Message Interpretation
AGTReadyNextItem	Request for no further work pending. Command is disabled.
Error text	E28904: Request for no further work is pending.

**E28905**

Source	Message Interpretation
--------	------------------------

Source	Message Interpretation
AGTReadyNextItem	Request to transfer to another job is active. Retry after job transfer is complete.
Error text	E28905: Request to transfer to another job is active.

**E28906**

Source	Message Interpretation
AGTCallNotify	The agent is not ready for next customer record. Call notification event is inappropriate.
AGTPreviewRecord	The agent is not ready for next customer record. Preview record event is inappropriate.
Error text	E28906: Not ready for next customer record.

**E28907**

Source	Message Interpretation
AGTManagedCall	Attached job is not a Managed Dialing job. Command is only available for Managed Dialing jobs.
Error text	E28907: Not a Managed Dialing job.

**E28908**

Source	Message Interpretation
AGTManagedCall	Agent is not previewing a customer record. Retry after receiving a record to preview.
Error text	E28908: No open customer record.

**E28909**

Source	Message Interpretation
--------	------------------------



Source	Message Interpretation
AGTManagedCall	Managed call is already complete. The AGTManagedCall command is already being executed. Might occur when agent executes the command just as the time-out elapses.
Error text	E28909: Managed dialing call is already complete.

**E28910**

Source	Message Interpretation
AGTManagedCall	Managed call already canceled or complete. Either an AGTFinishedItem, AGTReleaseLine, or AGTManagedCall command is already executing. Might occur if agent mistakenly executes a line release rather than a finished item or if the finished item executes just as the time-out elapses.
Error text	E28910: Managed dialing call is cancelled or complete.

**E28911**

Source	Message Interpretation
AGTManagedCall	Managed call already canceled. AGTFinishedItem is already executing.
Error text	E28911: Managed dialing call is canceled.

**E28912**

Source	Message Interpretation
AGTReadField	Customer record is not available. The agent is not working with a customer record.
Error text	E28912: Customer record is not available for update.

**E28913**

Source	Message Interpretation
--------	------------------------

Source	Message Interpretation
AGTDetachJob	No job is attached. There is nothing to detach.
Error text	E28913: There is no attached job to detach.

**E28914**

Source	Message Interpretation
AGTDetachJob	Agent is still available for work on the job and calls are still being routed to the agent headset. Execute AGTNoFurtherWork and retry.
Error text	E28914: Still available for work on the job.

**E28915**

Source	Message Interpretation
AGTDetachJob	Not logged out of the job yet. The AGTNoFurtherWork command is executing. Wait for a completion message from Proactive Contact and retry.
Error text	E28915: Not logged out of job.

**E28916**

Source	Message Interpretation
AGTLogoff	There is a job attached. Detach the job and retry.
Error text	E28916: Job attached. Detach job and retry.

**E28917**

Source	Message Interpretation
AGTJobTransLink	No job is attached. There is no current job that could have a job linked to it.

Source	Message Interpretation
AGTJobTransRequest	No job is attached. The transfer request is inappropriate.
AGTNoFurtherWork	No job is attached; nothing to log out of.
AGTSystemError	E28917: No job attached. Job linking is not available.
Error text	E28917: No job attached. Job linking is not available.

**E28918**

Source	Message Interpretation
AGTNoFurtherWork	Not available for work on the job. Agent is not logged in to a job; nothing to log out of.
Error text	E28918: Not available for work on the job.

**E28919**

Source	Message Interpretation
AGTFinishedItem	The agent is not currently working with a customer record. There is no customer record to release.
Error text	E28919: No active customer record to release.

**E28920 § <HeadsetID>†**

Source	Message Interpretation
AGTConnHeadset	<HeadsetID> is not in the reserved list. Retry the command with a valid reserved headset ID.
Error text	E28920: Headset ID is not found in reserved list - %s.

**E28921**

Source	Message Interpretation
--------	------------------------

Source	Message Interpretation
AGTSystemError	Fatal error - terminating. Proactive Contact has experienced a critical internal error and is shutting down.
Error text	E28921: FATAL ERROR. AGENT PROCESS (agent binary) ENDING.

**E28922**

Source	Message Interpretation
AGTReserveHeadset	No reserve headset ID request pending. Might indicate an Proactive Contact system problem.
Error text	E28922: No reserve headset ID request pending. Retry command with headset ID.

**E28923 § <HeadsetID>†**

Source	Message Interpretation
AGTReserveHeadset	The requested headset ID is reserved.
Error text	E28923: Headset ID %s is already reserved.

**E28924**

Source	Message Interpretation
AGTSystemError	Must sign in system first. Must execute AGTLogon before any other command.
Error text	E28924: Must log into system first.

**E28925**

Source	Message Interpretation
AGTLogon	This agent application is already logged in to the system from this session. To change the <AgentName> execute AGTLogoff and retry.

Source	Message Interpretation
Error text	E28925: Already logged on to the system.

**E28926**

Source	Message Interpretation
AGTLogon	Invalid sign-on. Indicates that the login name or password is invalid.
Error text	E28926: Invalid logon.

**E28942**

Source	Message Interpretation
AGTMoFlashBlind AGTMoFlashSupv	Transfer job is not available. The <JobName> specified is not running, does not exist, or is not an inbound or blend job.
Error text	E28942: Transfer job is not available.

**E28946**

Source	Message Interpretation
AGTAvailWork	Agent not yet acquired. The agent is logged in as an Agent Blending agent, but Proactive Contact has not acquired the agent for outbound calling.
Error text	E28946: Predictive Blend agent is not acquired for outbound calls.

**E28947**

Source	Message Interpretation
AGTFinishedItem	Completion code is invalid. Retry command with a code returned by AGTListKeys for the current job.
Error text	E28947: Invalid completion code.

**E28950**

<b>Source</b>	<b>Message Interpretation</b>
AGTLogonAcd	The specified extension is not one of the ACD extensions defined on Proactive Contact.
Error text	E28950: Extension not a valid ACD extension.

**E28951**

<b>Source</b>	<b>Message Interpretation</b>
AGTLogonAcd	The specified extension is in use by another agent.
Error text	E28951: Extension is in use by another agent.

**E28952**

<b>Source</b>	<b>Message Interpretation</b>
AGTLogonAcd	Cannot add another agent; the maximum number of ACD agents are already logged in.
Error text	E28952: Cannot add agent. Maximum number of ACD agents logged on.

**E28953**

<b>Source</b>	<b>Message Interpretation</b>
AGTLogonAcd	Could not read the file containing the ACD extensions. Indicates an Proactive Contact system problem.
Error text	E28953: Cannot read file containing ACD extensions.

**E28954**

<b>Source</b>	<b>Message Interpretation</b>
AGTLogonAcd	Duplicate login - Please try again. Indicates that the agent login name is already logged in to dispatcher.

Source	Message Interpretation
Error text	E28954: Duplicate login. Please try again.

**E28955**

Source	Message Interpretation
AGTLogonAcd	The Agent Blending dispatcher process is not running.
Error text	E28955: Predictive Blend dispatcher process is not running on Proactive Contact.

**E28956**

Source	Message Interpretation
AGTLogonAcd	ACD login error. Dispatcher returned an “unknown” response to the agent application.
Error text	E28956: Unknown ACD logon error.

**E28964**

Source	Message Interpretation
AGTReadyNextItem AGTManagedCall	The agent phone is busy.
Error text	E28964: Agent phone is busy. Release phone line before proceeding.

**E28965**

Source	Message Interpretation
AGTReadyNextItem AGTManagedCall	CTI link is down.
Error text	E28965: Softdialer link is down. Please wait until the link is up.

**E28967**

Source	Message Interpretation
AGTNoFurtherWork	An agent attempted to log out of an agent-owned recall without logging out of the job from which the agent transferred.
Error text	E28967: An agent is not allowed to logoff.

**E28985**

Source	Message Interpretation
AGTMoFlashBlind AGTMoFlashSupv	This error code appears when the customer is not online. This can also occur when the customer hangup the line or the line is released.
Error text	Customer is not online.

**E28812**

Source	Message Interpretation
AGTLogon	<AgentName> is logged in to another application process. The agent might be logged in at another location. Retry login with a different <AgentName>.
Error text	E28812: Agent %s already logged on. Access is denied.

**E29000**

Source	Message Interpretation
AGTAvailWork	Only Managed agents might join this job. The agent application agent type (work class) does not match the job type. Reset the agent type to M and retry.
Error text	E29000: Agent type is not M. Cannot join Managed Dialing job.



**E29203**

Source	Message Interpretation
AGTSysError	Could not attach slot.
Error text	E29203: System error. Could not attach slot.

**E29206 § <unit>†**

Source	Message Interpretation
AGTAttachJob	Cannot attach the <unit> segment of shared memory that Proactive Contact is using for the job. Most jobs use the same unit of shared memory, defined in a configuration file on the Proactive Contact system. This message indicates a problem with the Proactive Contact configuration.
Error text	E29206: System error. Failed to attach %s segment of shared memory.

**E29950**

Source	Message Interpretation
AGTHoldCall AGTUnholdCall AGTManualCall AGTHookFlashLine AGTTransferCall AGTDialDigit AGTAdjustHeadset	This command is not available on an Avaya Proactive with CTI system.
Error text	E29950: Feature not available in Softdialer Mode

**E29952**

Source	Message Interpretation
AGTAvailWork	This error code appears when there is a licensing issue. If the system is not able to contact the enforcer service.
ErrorText	Failed to join job. Please contact system administrator.

**E29953**

Source	Message Interpretation
AGTFindFirstRecord AGTFindNextRecord AGTCancelSearch	Preview search functionality is not enabled.
ErrorText	PVRS not enabled

**E29954**

Source	Message Interpretation
AGTFindNextRecord AGTCancelSearch	First initiate the search operation before executing these commands.
ErrorText	Search not in progress

**E29955**

Source	Message Interpretation
AGTFindFirstRecord	Search key value is not provided
ErrorText	Search Value Not Entered

**E29956**

Source	Message Interpretation
AGTFindFirstRecord	Search operation is already initiated.
ErrorText	Search Already in Progress

**E29959**

Source	Message Interpretation
AGTSetUnitt	Exceeded the maximum no. of units that can be selected for a multi-unit selection.

Source	Message Interpretation
ErrorText	Number of selected units exceeded maximum allowed limit.

**E50100**

Source	Message Interpretation
AGTLogon	The maximum number of agents allowed by Agent_count are already logged in to the system.
Error text	E50100: Exceeded the maximum number of agents allowed.

**E50611 § <HeadsetID>‡**

Source	Message Interpretation
AGTReserveHeadset	The requested headset ID is in use.
Error text	E50611: Headset ID is already reserved.

**E50612 § <HeadsetID>‡**

Source	Message Interpretation
AGTReserveHeadset	No more headsets are permitted on the system.
Error text	E50612: No more headsets permitted on the system.

**E50613 § <HeadsetID>‡**

Source	Message Interpretation
AGTReserveHeadset	Failure to access the headset ID table. Might indicate a permissions problem for the headset ID table file on the Proactive Contact system, a corrupt headset ID table file, or an Proactive Contact system problem.
Error text	E50613: Failed to access the headset ID file.

**E58006**

Source	Message Interpretation
AGTSetTenant AGTListTenants	Failed to load the user and tenant mapping. This might indicate that there is problem in usermap configuration file.
Error text	E58018 : Failed to load tenant user mapping.

**E58007**

Source	Message Interpretation
AGTSetTenant	Tenant name is not valid or there might be problem with tenant configuration file.
Error text	E58018 : Unable to get tenant id for tenant name.

**E58018**

Source	Message Interpretation
AGTSetTenant	Tenant is already set.
Error text	E58018 : Tenant is already set.

**E58021**

Source	Message Interpretation
AGTAttachJob	Tenat is not set. Tenant must be set before joining job.
Error text	E58021 : Tenant is not set, set tenant first.

**E58022**

Source	Message Interpretation
AGTSetTenant	Invalid tenant name.
Error text	E58018 : Tenant name is invalid or the use does not belong to the tenant.

**E70000**

Source	Message Interpretation
Generic	Incorrect number of arguments. The command issued from the agent application contained the wrong number of arguments.
Error text	E70000: Incorrect number of arguments.

**E70001**

Source	Message Interpretation
Generic	Incorrect message type (not Data, Pending, Error, Response, Busy, or Notify). This code might indicate that the agent application message headers are not formatting correctly.
Error text	E70001: Incorrect message type.

**E70002, <message>**

Source	Message Interpretation
Generic	Pending request timed out waiting for <message>. The <message> is a text string containing the command that timed out and the response that the agent was expecting from Proactive Contact. This code might indicate an Proactive Contact system problem.
Error text	E70002: Timed out waiting for %s.

**E70003**

Source	Message Interpretation
AGTListJobs	Unknown job type. Retry with a different job type.
Error text	E70003: Unknown job type.

**E70006**

Source	Message Interpretation
AGTAvailWork	Unit ID value not selected. The agent is logged in to a Unit Work List job but has not selected a unit ID value to work with. Execute AGTSetUnit and retry.
Error text	E70006: Need to select work unit.

**E70007**

Source	Message Interpretation
AGTMoFlashBlind	Cannot transfer an inbound call. AGTMoFlashBlind can only transfer outbound calls.
AGTMoFlashSupv	Cannot transfer an inbound call. AGTMoFlashSupv can only transfer outbound calls.
Error text	E70007: Cannot transfer an inbound call.

**E70008**

Source	Message Interpretation
AGTMoFlashBlind AGTMoFlashSupv	Must specify a transfer job. There is no default transfer job configured on Proactive Contact, so you must include a transfer <JobName> with a command.
Error text	E70008: Must specify a transfer job.

**E70009 <process name>**

Source	Message Interpretation
AGTMoFlashBlind AGTMoFlashSupv	Unable to send a message to <process name>. Internal Proactive Contact system error; agent cannot send messages.
Error text	E70009: Unable to send the message.

**E70010**

Source	Message Interpretation
AGTMoFlashSupv	Conference is in progress. The agent executed the command a third time after the conference call has begun.
Error text	E70010: Conference call is already in progress.

**E70011**

Source	Message Interpretation
AGTLogonAcd	Agent Blending is not available on this Proactive Contact system. This system cannot work with ACD agents.
Error text	E70011: Predictive Blend is not available on this system.

**E70012**

Source	Message Interpretation
AGTSetPassword	The agent's current password is no longer valid. Enter a new password to continue the log in process.
Error text	E70012: Password has expired, must be changed.

**E70013**

Source	Message Interpretation
--------	------------------------

Source	Message Interpretation
AGTSetPassword	Proactive Contact is not configured to allow an agent to change his or her password.
Error text	E70013: Password can not be changed, change limit not expired.

**E70014**

Source	Message Interpretation
AGTSetPassword	Proactive Contact is configured with a locked password file. The agent cannot change his or her password.
Error text	E70014: Password can not be changed, password file locked.

**E70015**

Source	Message Interpretation
AGTSetPassword	The agent does not have access privileges to set a password*. Contact your system administrator to reset a password.
Error text	E70015: Unable to become root privilege for setting password.

**E70016**

Source	Message Interpretation
AGTSetPassword	The password the agent entered is not correct. Retry using a different password that follows the password* rules.
Error text	E70016: Original password is invalid.

**E70017**

Source	Message Interpretation
--------	------------------------



Source	Message Interpretation
AGTSetPassword	The new password entry is incorrect. Retry using a different password that follows the password* rules.
Error text	E70017: New password entered is invalid.

See [AGTSetPassword](#) on page 201 for password rules.

## Data Messages

Message Number	Command(s)	Message Interpretation
M00000	Generic  Error text	Complete.  M00000: Successful completion.
M00001 § <CompCode>, <Description>, <ScriptLabel> § <CompCode>, <Description>, <ScriptLabel> §...‡	AGTListKeys	Data message. Completion codes are 3-digit numbers. Customers define call completion codes during specification of their Proactive Contact system. The descriptions are from the Proactive Contact compcode.cfg file. Telephone script labels are found in the Proactive Contact telephny.spt file. The system associates the labels with the respective call completion codes.
M00001§ <FieldName>, <FieldLength>, <FieldType>,F § <FieldName>, <FieldLength>, <FieldType>, F...‡	AGTListDataFields  AGTListCallFields	Data message. The <FieldType> is C (character), N (numeric), \$(currency), D (date) or T (time). The F value is a placeholder for future use.
M00001 § <CallStatus>‡	AGTManagedCall	Data message. <CallStatus> should always be (CONNECT).
M00001 § <JobType>, <JobName>, <Status> § <JobType>, <JobName>, <Status>...‡	AGTListJobs	Data message containing requested job information. Job status appears as a single character, I for inactive or A for active.
M00001 § <Message>‡	AGTReceiveMessage	Data message. The text of the message the supervisor sent.
M00001 § <UnitID>, <UnitID>... ‡	AGTListUnits	Data message. There are as many data elements as there are unit ID values on the job.

Message Number	Command(s)	Message Interpretation
M00001\$ <FieldName>, <FieldType>, <FieldLength>, <FieldValue> ‡	AGTReadField	Data message. <FieldName> is the name specified in the command. <FieldType> is the type of data in the field: A (alphanumeric), N (numeric), D (date), T (time), or \$(currency). <FieldLength> is the number of characters in the field. <FieldValue> is the value in the current customer record.
M00001\$ <Ear> § <Mouth>‡	AGTGetHeadsetVol	Data message. The current volume settings for the headset earphone and microphone. Range for settings is from one to eight.
M00001\$ <FieldName>, <FieldData> § <FieldName>, <FieldData> ...‡	AGTCallNotify  AGTPreviewRecord	Additional fields data message.
M00001\$ <Format> § <Phones>‡	AGTListCallbackFmt	Data message. <Format> is the date format used on Proactive Contact. <Phones> is the number of phones in the customer record available for recall. This provides a range (from 1 to <Phones>) of index values to use with AGTSetCallback.
M00001\$ <ListName1> § <ListName2> ...‡	AGTListCallLists	Data message. The data message includes as many data segments as there are calling lists on the system.
M00001\$ <OpMesg> § <CallType> § <NotifyFieldName>, <FieldData>‡	AGTPreviewRecord	Initial preview data message.
M00001\$ <OpMesg> [*<WaitMsg>] § <CallType> § <NotifyFieldName>, <NotifyFieldData>‡	AGTCallNotify	Initial notification data message.

Message Number	Command(s)	Message Interpretation
M00001\$ <Message No.>,<Records Found>,<Current Record> ‡	AGTFindFirstRecord AGTFindNextRecord	Data Message. <Message No.> is the message number, <Records Found> is the total searched records and <Current Record> is the current record index that is going to be displayed
M00001\$ <Records Found> records found, <Current Record> is being displayed\$ MANAGED\$ <Notify Key Field Name>, <Value> ‡	AGTFindFirstRecord AGTFindNextRecord	<Records Found> is the total searched records found, <Current Record> is the current record index, <Notify Key Field Name> is field name that comes with the notification and its <value>
M00001\$ <Field Name>, <Value> ‡	AGTFindFirstRecord AGTFindNextRecord	<Field Name> is the fields that are set to get data at the time of call and <value> is the field value
M00001 \$ <TenantName1>, <TenantName2>‡	AGTListTenants	Data message. The data message includes tenant names for agent.
M00001	Error text	M00001: Data message.

## Pending Messages

In the information that follows, each message number has its own heading.

### S28814

Commands	Message Interpretation
AGTAdjustHeadset	Transfer is in progress. The last call is being transferred. Retry the command after connecting to the next call.
AGTDialDigit AGTHookflashLine	The transfer for the last call is in progress. Retry after the transfer is complete.
AGTGetHeadsetVol	A transfer is in progress. The customer call is still being transferred to the agent. Retry after connecting to the call.
AGTHoldCall	Transfer is in progress. Proactive Contact cannot place the call on hold.
AGTManualCall	Transfer in progress. The last call is being transferred. Retry after transfer complete.
AGTMoFlashBlind AGTMoFlashSupv	Trunk-to-trunk transfer is already in progress. Might mean the agent previously executed AGTTransferCall.
AGTUnholdCall	Transfer is in progress. The call is being transferred.
Error text	S28814: Transfer is in progress.

### S28833

Commands	Message Interpretation
Generic	Pending.
Error text	S28833: Request is pending.

### S28971

Commands	Message Interpretation
AGTlicbFeNotif	Standby to take outbound calls... Acquisition to outbound is pending.

Commands	Message Interpretation
Error text	S28971: Standby to take outbound calls...

**S28972**

Commands	Message Interpretation
AGTlicbFeNotif	Standby to take inbound calls... Release to inbound is pending.
Error text	S28972: Standby to take inbound calls...

**S28833**

Commands	Message Interpretation
Generic	Pending.
Error text	S28833: Request is pending.

**S70000, <JobName>**

Commands	Message Interpretation
AGTListState	Agent is on a call. Working on <JobName>, currently working with a customer record.
Error text	S70000: Working on job %s. An agent is using a customer record.

**S70001, <JobName>**

Commands	Message Interpretation
AGTListState	Agent is ready for a call. Working on <JobName>, currently waiting for next customer record.
Error text	S70001: Working on job %s. Waiting for the next customer record.

**S70002, <JobName>**

Commands	Message Interpretation
AGTListState	Agent has joined (logged in to) a job. Working on <JobName>, currently available for work but not ready for next item.
Error text	S70002: Working on job %s. Not ready for next customer record.

**S70003, <JobName>**

Commands	Message Interpretation
AGTListState	Agent has selected a job to work with. Attached to <JobName>, but not yet available for work.
Error text	S70003: Job %s attached. Not yet ready for work.

**S70004**

Commands	Message Interpretation
AGTListState	Agent logged in to Proactive Contact. Agent is idle, not yet attached to a job.
Error text	S70004: No job attached. Agent is idle.

**S28996**

Commands	Message Interpretation
AGTJobMode	When agent joins manual mode enabled job.

**S28997**

Commands	Message Interpretation
AGTJobMode	When agent joins Preview empty record enabled job.

## Appendix D: Moagent32.dll Interfaces

This appendix details the interface properties, events, and methods associated with the Moagent32.dll. For information on connecting to the DLL from your client application using methods, setting properties, or receiving events (from Proactive Contact), see [Using Proactive Contact Agent API DLL](#) on page 52.

This section contains the following topics:

- [IServerStartup Interface](#)
- [IConfigure Interface](#)
- [IMoagent Interface](#)

### IServerStartup Interface

This interface is a COM interface. Client applications access Moagent32.dll through the properties in this interface.

#### Properties

Property	Description
PropIConfig () As IConfigure	Returns a pointer to the IConfigure interface.
PropNexus() As IMoagent	Returns a pointer to the IMoagent interface.
PropWinHwnd	Provides Moagent32.dll with a client window handle for post-message event notification. (Use for PowerBuilder client applications only.)
PropWinUserMsg	Provides Moagent32.dll with a client window handle for post-message event notification. (Use for PowerBuilder client applications only.)

#### Events

None.

#### Methods

None.

### IConfigure Interface

This interface is public usable.



## Properties

Property	Description
SetCreateStatFile	When set to "True," Moagent32.dll creates an agent statistics file.
SetStatFileName	When set to "True," Moagent32.dll writes statistics to the file name you specify (can include path).
SetErrFileName	When set to "True," Moagent32.dll creates an error file consisting of the last N errors (identified by SetNumOfLstErrs).
SetNumOfLstErrs	Specifies the number of errors to include in the errors display and error file (first in, first out).
SetCreateLogFile	When set to "True," Moagent32.dll creates a message transaction log file (identified by SetLogFileName).
SetLogFileName	Specifies the name of the message transaction log file.
SetLogonRecovery	When set to "True," Moagent32.dll detects an invalid user ID and/or password entry and displays a dialog box prompting the user to reenter a valid user ID and/or password.
SetReserveConnHeadSet	When set to "True," Moagent32.dll automatically reserves and connects a headset after login if a headset value is given.
SetLogonTimeout	Specifies how long Moagent32.dll waits (in seconds) before sending a message advising that the Proactive Contact login attempt timed out. The timeout range is between 2 and 200 seconds.
SetAvayaMosaixTimeout	Specifies how long Moagent32.dll waits (in seconds) before sending a message advising that the Proactive Contact command failed. The timeout range is between 2 and 200 seconds.

Property	Description
SetUseDllDlls	<p>When set to “True,” Moagent32.dll makes Dialog Boxes available (see Dialog Boxes, later in this appendix). Use the following commands without arguments, for example AttachJob( ), to display a dialog box that prompts a user for data:</p> <ul style="list-style-type: none"> <li>○ AttachJob</li> <li>○ FinishedItem</li> <li>○ SetWorkClass</li> <li>○ AdjustHeadset</li> <li>○ DialDigit</li> <li>○ SendMessage</li> <li>○ SetCallback</li> <li>○ SetPassword</li> <li>○ SetUnit</li> <li>○ TransferCall</li> </ul>

## Events

None.

## Methods

None.

## IMoagent Interface

This interface is public usable. See [Using Proactive Contact Agent API DLL](#) on page 52 for information on using the events and methods associated with the IMoagent interface in your client application. See [Commands and Notification Events](#) for descriptions of the Proactive Contact API commands associated with the methods listed below.

## Properties

None.

## Event

Avaya MosaixEvent (ErrFlag As Boolean, NotifyType As String,  
Avaya MosaixDataPacket As String, ErrCode As String, ErrText As String)

## Methods

The following methods have no arguments that do not return data:

- AvailWork (ErrCode,ErrText)
- ConnectHeadset (ErrCode,ErrText)
- DetachJob (ErrCode,ErrText)
- DisconnHeadset (ErrCode,ErrText)
- DoNotCall (ErrCode, ErrText)
- EchoOff (ErrCode,ErrText)
- EchoOn (ErrCode,ErrText)
- FreeHeadset (ErrCode,ErrText)
- HangupCall (ErrCode,ErrText)
- HoldCall (ErrCode,ErrText)
- LogloStart (ErrCode,ErrText)
- LogloStop (ErrCode,ErrText)
- Logoff (ErrCode,ErrText)
- LogoffAcd (ErrCode,ErrText)
- NoFurtherWork (ErrCode,ErrText)
- ReadyNextItem (ErrCode,ErrText)
- UnholdCall (ErrCode,ErrText)
- CancelSearch (ErrCode As String, ErrText As String)
- GetJobMode()
- GetClickToDial()

The following methods have no arguments that return data:

- ListCallbackFmt (ErrCode,ErrText)
- GetHeadsetVol (ErrCode,ErrText)
- ListCallLists (ErrCode,ErrText)
- ListKeys (ErrCode,ErrText)
- ListState (ErrCode,ErrText)
- ListTenants(ErrCode,ErrText)
- ListUnits (ErrCode,ErrText)
- ManagedCall (ErrCode,ErrText)

The following methods have arguments that do not return data:

- AdjustHeadset (EarMouth,Volume,ErrCode,ErrText)

- AttachJob (JobName,ErrCode,ErrText)
- ClearDataSet (WorkClass,ErrCode,ErrText)
- DialDigit (Digit, ,ErrCode,ErrText)
- DumpData (FileName,ErrCode,ErrText)
- FinishedItem (CompCode,ErrCode,ErrText)
- HookflashLine (PhoneNumber,ErrCode,ErrText)
- Logon (ServerName, PortNumber, UserID, PassWrd, Headset,ErrCode,ErrText)
- LogonAcd (Extension,PbxID,ErrCode,ErrText)
- ManualCall (PhoneNumber,ErrCode,ErrText)
- MoFlashBlind (JobName,ErrCode,ErrText)
- MoFlashSupv (JobName,ErrCode,ErrText)
- ReleaseLine (ScriptLabel (or MessageNo),ErrCode,ErrText) ScriptLabel/ Message optional.
- ReserveHeadset (Headset ID,ErrCode,ErrText)
- SendMessage (Message,ErrCode,ErrText)
- SetCallback (CallBackDate, Time, PhoneIndx, RefName, ManualPhNum, ErrCode,ErrText)
- SetDataField (WorkClass,FieldName,ErrCode,ErrText)
- SetNotifyKeyField (WorkClass,FieldName,ErrCode,ErrText)
- SetPassword (UserID, PresentPW, NewPW, ErrCode, ErrText)
- SetTenant (TenantName, ErrCode, ErrText)
- SetUnit (UnitID,ErrCode,ErrText)
- SetWorkClass (WorkClassID,ErrCode,ErrText)
- TransferCall (PhoneNumber,ErrCode,ErrText)
- UpdateField (WorkClass,Fieldame,NewValue,ErrCode,ErrText)

The following methods have arguments that return data:

- ListCallFields (ListName,ErrCode,ErrText)
- ListDataFields (WorkClass,ErrCode,ErrText)
- ListJobs (JobType,ErrCode,ErrText)
- ReadField (WorkClass,FieldName,ErrCode,ErrText)
- ManualPhone(PhoneNumber, FormatedPhoneNumber, ErrCode,ErrText)
- FindFirstRecord (sKeyVal, sRecordData, ErrCode, ErrText)
- FindNextRecord (sNext, sRecordData, ErrCode, ErrText)

The `GetAgentState ( )` method is specific to `Moagent32.dll`. Calling this method from your client application causes `Moagent32.dll` to display a dialog box with current system state information (for the client machine).

## Appendix E: Data Parameters

This appendix lists data parameters in alphabetical order.

---

The agent binary returns some of the same parameters to the agent application in data records or as part of the Proactive Contact system error messages.

Data Parameter	Description
<AgentName>	The user name of the calling agent. Use a maximum of 19 characters, 7-bit USASCII character set. Do not include embedded spaces.
<CallType>	Identifies the call source. If Proactive Contact places the call, it is OUTBOUND. If the customer places the call, it is INBOUND. If it is a Managed Dialing job call, the type is MANAGED.
<ClassID>	The code for the agent type. One character, alphabetic, case sensitive. Agent type codes are I (inbound), O (outbound), B (blend), P (Person to Person), and M (Managed).
<CompCode>	Call Completion Code. The 3-digit numeric code to place in the customer record. It indicates the results of the call. The code must be in the list returned by the AGTListKeys command.
<Date>	The date to place the recall. CCYY/MM/DD, MM/DD/ CCYY, or DD/MM/CCYY. Must match the format returned by AGTListCallbackFmt.
<Digit>	The digit for Proactive Contact to call. Use one character: a single number (0 - 9), *, or #. If the Proactive Contact master.cfg file contains values for the DIAL_STAR and DIAL_POUND variables, Proactive Contact substitutes those values for the normal * or # tones.
<EarMouth>	The headset volume setting to change: E (ear) or M (mouth).
<ErrorMessage>	The Proactive Contact error message number and any data or information that is part of that message.
<Extension>	The ACD extension to associate with the agent's telephone. Use a numeric value, up to 20 characters.
<FieldName>	A field name from the calling list, as shown by AGTListDataFields. Up to 19 characters, alphanumeric, case sensitive, no embedded spaces.

Data Parameter	Description
<FieldData>	Value of data field from the customer record.
<FileName>	The file name for the dump file. To use AGTDumpData repeatedly in a single client session without overwriting the dump file, vary the file name. Use 7-bit USASCII, no embedded spaces or special characters. The <FileName> parameter is case sensitive.
<HeadsetID>	Identifies the extension for the agent's headset. Up to 14 numeric characters, no special characters or embedded spaces.
<JobName>	The name of a job on Proactive Contact. Use up to 19 characters, 7-bit USASCII. The <JobName> is case sensitive and might not include special characters or embedded spaces.
<JobType>	Type of job: all (A), inbound (I), blend (B), Managed Dialing (M), or outbound (includes Unit Work List and Sales Verification) (O). Use one alphabetic, case sensitive, character.
<ListName>	The name of the calling list. Use alphanumeric values, 7-bit USASCII. Calling list names are case sensitive.
<ListType>	Type of calling list used by the attached job: inbound (I) or outbound (O). Blend jobs use both I and O calling lists. Inbound jobs use type I calling lists. Outbound, Managed Dialing, Unit Work List, and Sales Verification jobs use type O calling lists.
<Message>	The text of the message from the supervisor or to display on the supervisor's screen. The message might be up to 79 characters long. Embedded spaces and special characters are allowed.
<MessageNo>	Value is the message number to play to the customer. These numbers must appear in the voicemsg.cfg file on the Proactive Contact system. Use up to three digits, ranging from 1 to 255.
<NewValue>	A value to insert in the field. The value must fall within the parameters for the field returned by AGTListDataFields or AGTReadField, with regard to length and type of value (Alphanumeric, Numeric, Date, Time or Currency).
<NotifyFieldData>	The notification key field value in the customer's record.
<NotifyFieldName>	The calling list field name requested by AGTSetNotifyKeyField.

Data Parameter	Description
<OpMesg>	Messages from call or preview notification containing field information from the customer record. The field choice is configured in alljobs.dat on the Proactive Contact system. The default field is NAME. If the call is a voice and data transfer from another agent, the <OpMesg> is "TRANSFER CALL." If the notification is a Managed Dialing preview record, the text string "Preview" always follows the information.
<Password>	Enter the agent's password. Use two to eight characters, 7-bit USASCII character set. Do not include embedded spaces.
<PBX ID>	The identification of the PBX (switch) used for Agent Blending. Default value is 1. Value given here must match the DBKGROUP parameter in the Proactive Contact master.cfg file.
<PhoneIndex>	Index of the Proactive Contact phone field to use for the recall. Numeric, with a value between 1 and 4. The choice of digits determine which phone index to use. A 1 corresponds to the PHONE1, a 2 to the PHONE2 field, and so on. The digit must fall in the range returned by AGTListCallbackFmt.
<PhoneNumber>	The phone number to call. Numeric, up to 43 characters, no embedded spaces or special characters. If the number is a complete telephone number rather than an extension, the format must match the standard format in the phonefmt.cfg configuration file on the Proactive Contact system.
<RecallName>	This is the customer name field to contact during the recall. There might be a default calling list field name to use for recalls specified in the job's *.fdict file. This parameter is optional.
<RecallNumber>	This is the telephone number to use for the recall (instead of the Phone Index specification that can be entered or left blank.) There might be a default calling list field number to use for recalls specified in the RECALLNUMBER parameter of the job's fdict file. This parameter is optional. This field is numeric, up to 43 characters, no embedded spaces or special characters. If the number is a complete telephone number rather than an extension, the format must match the standard format in the phone.cfg configuration file on the Proactive Contact system.
<ScriptLabel>	Optional parameter. Value is the script label to execute in the calling script (telephny.spt). Use AGTListKeys to get available labels. Use up to 15 characters, 7-bit USASCII case sensitive text with no embedded spaces or special characters.



Data Parameter	Description
<Time>	The time of day when the recall should take place. Four or five characters in one of the forms: HHMM, HHMM[A or P], or HHMM+. The time represented is a 24-hour clock (HHMM), 12-hour clock (HHMM[A or P] where A=AM, P=PM), or incremental time (HHMM+) from current time.
<UnitID>	The unit ID value selected by the agent.
<Volume>	The numeric value for the headset volume setting. Whole numbers only, from 1 to 8.
<WaitMsg>	If present, an asterisk (*) separates <WaitMsg> from <OpMesg>. <WaitMsg> is a text string defined in the Proactive Contact job file indicating how long the customer has been on hold. <WaitMsg> might be up to 30 characters long. Sometimes the message is for the agent's information only ("5-10 seconds"), sometimes it is for the agent to read to the customer ("We're sorry you had to hold").
<TenantName>	The tenant name of the calling agent. Use a maximum of 20 characters. Do not include embedded spaces.

## Appendix F: Use the Agent DDE Interface

The Agent application exposes an interface that allows other windows applications to access some information about the current call. The following operations are supported:

- Throw an event when a call is received
- Throw an event when preview data is received
- Throw an event when a preview call is connected
- Throw an event when the agent changes customer data
- Expose the customer data field names and values
- Change the value of the customer data on the Agent's screen

### Events and Methods

This interface is implemented in DDE. DDE requires three attributes:

- Application Name
- Topic Name
- Item Name.

The Agent Application and Topic Names are always AgentDDE and CallData, respectively. Below is the Item Name for each event or method, and a description of the data that they expose.

### CallType

Indicates whether the call is inbound or outbound. It contains an I or an O respectively. This link is updated when a new call arrives; it is not updated when a preview call arrives because preview calls are always outbound.

### CallFields

Lists all of the customer data for the call in a single string. Each field is delimited by a dash, and the name and value are delimited by a comma. This is an example:

```
NAME2,JOHN DOE ACCTNUM,5300292160909890 PHONE1,2037531065
```

---

The Agent does not call "AGTSetDataField" on every field in the calling list. It calls it only for fields that are defined in a screen (for performance reasons). Therefore, only fields that are defined in a screen are exposed through this method.

### CallNotify

Contains the data associated with the "CallNotify" field. (The CallNotify field is a special field that is defined in the Job file. It is normally the account number.)

## PreviewFields

Contains the same data as the CallFields event, but it is thrown when the data for a preview call arrives.

## PreviewNotify

Contains the same data as the CallNotify event, but it is thrown when the data for a preview call arrives.

## ManagedCall

When the voice connection for a preview call arrives, the value of this item changes to "HAVE CALL."

## FieldChanged

Whenever a customer data record is successfully updated on the dialer, this event exposes the name of the field and its new value. A comma is used to separate the two.

---

This is not thrown when the agent types in the change, but only when the change is saved to the dialer. Fields are saved in batches, so this event will be thrown several times in succession (once for each changed field).

There are three ways for the Agent to perform a Save:

- by pressing the Save button
- by switching to a different screen
- by finishing the work

## PushField

This is the method. It allows another application to change the value associated with one of the customer fields. The DDE execute function must be used to send a string to the agent. The string must be a comma-delimited name-value pair with the first string being the name of an existing field, and the second string being the new value of the field. For example:

NAME2, Joe

## Command Line Flags

The following command line flags can be used with the Agent:

Command line flag	Description
-------------------	-------------

Command line flag	Description
-r read-only	The Agent will not write any configuration back to the ini file. Note: this will disable features such as remembering release keys for the shortcut bar, and remembering the position and size of the window.
-c [filename]	Explicitly points the agent to an ini file in a different directory. The default is a file named PCAgent.ini in the PCAgent directory.
-l [filename]	Explicitly specifies the path and filename of the log file.
-sl [filename]	Explicitly specifies the path and filename of the socket log file.
-nc	No configuration. Tells the agent that there is no ini file, and don't look for one. This will disable all features that rely on saving information in the ini file. Also, you'll need to use the -d flag to supply a list of dialers to choose from. Normally, this list is in the ini file, but since there is no ini file, you need to specify them on the command line. This flag also disables both logs. If you want logs, you need to specify the -l and -sl flags.
-d [IP address]	Specifies a dialer's address to be placed in the list for the agent to choose from. Multiple -d flags can be used to put multiple entries in the list.

## **Appendix G: Avaya Proactive Contact 5.1.2/5.1.3 Agent API clients interacting with PC 5.2 Dialer**

Since Avaya Proactive Contact 5.1.2/5.1.3 Agent API client doesn't support TLSv1.1 and TLSv1.2, TLS configuration in 5.2 Dialer needs to be changed from default TLSv1.2 to SSLv23.

The TLS configuration can be changed in Dialer using sysadm CUI menu as below,

Administrator Menu->Administrative tasks->Configure SSL/TLS and Ciphers->Configure SSL/TLS

Mention the value of CORE\_SSL\_METHOD to SSLv23 and restart the Dialer.

# Index

- acquired phone numbers, 14
- Agent API messages, 20
- Agent application, 17
- Agent application initiated messages, 21
- agent application session sample, 16, 34
- Agent applications, 17
- Agent binary, 17
- agent owned recall sample, 16, 47
- Agent states, 25
- Agent types, 11
- Agent\_state, 68
- AgentName, 284
- AGTAAjustHeadset, 70, 72, 113, 141
- AGTAvailWork, 78
- AGTCallNotify, 81
- AGTClearDataSet, 86
- AGTConnHeadset, 88
- AGTDetachJob, 90
- AGTDialDigit, 91
- AGTDisconnHeadset, 94
- AGTDoNotCall, 96
- AGTDumpData, 97
- AGTEchoOff, 101
- AGTEchoOn, 103
- AGTFinishedItem, 108
- AGTFreeHeadset, 110
- AGTGetHeadsetVol, 111
- AGTHangupCall, 115
- AGTHeadsetConnBroken, 116
- AGTHoldCall, 117
- AGTHookflashLine, 119
- AGTlicbAbort, 121
- AGTlicbFeNotif, 122
- AGTlicbOffline, 123
- AGTlicbOnline, 125
- AGTJobEnd, 126, 216
- AGTJobTransLink, 127
- AGTJobTransRequest, 129
- AGTListCallbackFmt, 130
- AGTListCallFields, 132
- AGTListCallLists, 134
- AGTListDataFields, 135
- AGTListJobs, 138
- AGTListKeys, 139
- AGTListState, 142
- AGTListUnits, 145
- AGTLogloStart, 146
- AGTLogloStop, 152
- AGTLogoff, 153
- AGTLogoffAcd, 154
- AGTLogon, 156
- AGTLogonAcd, 158
- AGTManagedCall, 161
- AGTManualCall, 163
- AGTMoFlashBlind, 167
- AGTMoFlashSupv, 171
- AGTNoFurtherWork, 175
- AGTPreviewRecord, 177
- AGTReadField, 180
- AGTReadyNextItem, 182
- AGTReceiveMessage, 184, 185
- AGTReleaseLine, 187
- AGTReserveHeadset, 189
- AGTSendMessage, 191
- AGTSetCallback, 192
- AGTSetDataField, 197
- AGTSetNotifyKeyField, 199
- AGTSetPassword, 201
- AGTSetUnit, 204
- AGTSetWorkClass, 207
- AGTSTART, 209
- AGTSystemError, 210
- AGTTransferCall, 212
- AGTUnholdCall, 214
- AGTUpdateField, 217
- AGTXferCustHangup, 219

## Avaya Proactive Contact Agent API 5.2

- AGTXferTrunkHangup, 220
- alternate phone numbers, 14
- API, 53
- Automate call handling, 19
- Avaya Proactive Contact server, 17
- Avaya Proactive Contact system error messages, 233
- Basic, 54
- binary, 17
- binary initiated messages, 16, 21
- Blend calling, 13
- C++, 55
- call handling, 19
- Calling alternate phone numbers, 14
- calling lists:definition, 5
- calling lists:environment, 6
- calls, 14
- CallType, 284
- ClassID, 284
- Code samples, 59
- COM, 53
- Command message formats, 226
- Command messages, 20
- CompCode, 284
- customer records, 14
- customize, 67
- Data from agent application, 25
- Data from agent binary, 25
- Data messages, 20
- data separators, 25
- DataPacket, 281
- Date, 284
- default properties, 59
- Developer requirements, 52
- dialers:functions, 4
- dialers:multiple, 4
- dialog boxes, 61
- Dialog\_box\_text, 68
- Digit, 284
- EarMouth, 284
- Ending calls, 15
- ErrorMessage, 284
- Event, 281
- events, 57
- Exit Avaya Proactive Contact, 19
- Extension, 284
- FieldData, 285
- FieldName, 284
- FileName, 285
- formats, 16, 21
  - Data message formats:message formats:data, 228
  - Notification event message formats:message formats:notification event, 230
  - Response message formats:message formats:response, 228
- Handle calls, 12
- headers, 21
- headset, 19
- HeadsetID, 285
- headsets, 15
- IConfigure interface, 279
- IMoagent interface, 281
- Inbound calling, 13
- interface methods, 55
- interface:IConfigure, 279
- interface:IMoagent, 279, 281
- IServerStartup interface, 279
- job, 16, 29, 40, 45
- JobName, 285
- jobs, 9, 12
- JobType, 285
- Join a job, 19
- Join jobs, 12
- Leave a job, 15
- ListName, 285
- ListType, 285
- Log in to Avaya Proactive Contact, 11
- Log out, 19
- Log out of Avaya Proactive Contact, 15
- Logon, 67
- managed dialing job, 16, 40

## Avaya Proactive Contact Agent API 5.2

- manual calls, 14, 19
- Message, 285
- Message data separators, 25
- message formats, 226
- Message formats, 21
- Message headers, 21
- message scenarios, 29
- Message terminators, 25
- MessageNo, 285
- messages:Avaya Proactive Contact system error, 233
- Method syntax, 66
- methods, 55
- Methods, 282
- Moagent32 interface, 56, 66
- Moagent32.dll, 53, 57
- Moagent32.dll dialog boxes, 61
- Moagent32.ini settings, 67
- Moagent32.log, 34
- NewValue, 285
- Notification event messages, 20
- notification events, 57
- NotifyFieldData, 285
- NotifyFieldName, 285
- OpMesg, 286
- Outbound calling, 12
- Outbound job commands (by State), 28
- Outbound job message scenarios, 29
- parameters:data, 284
- Password, 286
- PBX ID, 286
- PhoneIndex, 286
- PhoneNumber, 286
- Placing manual calls, 14
- Power Builder, 54
- calling lists:processing, 5
- properties, 59
- PropIConfig, 279
- recall, 47
- RecallName, 286
- RecallNumber, 286
- recalls, 14
- records, 14
- requirements, 52
- Response messages, 20
- completion codes:list of, 222
- Sample agent application session, 34
- Sample agent owned recall, 47
- Sample managed dialing job, 40
- Sample unit work list job, 45
- samples, 59
- Scheduling recalls, 14
- ScriptLabel, 286
- security, 8
- separators, 25
- server, 17
- Server\_return\_codes, 68
- SetAvayaPDSTimeout, 280
- SetCreateLogFile, 280
- SetCreateStatFile, 280
- SetErrFileName, 280
- SetLogFileName, 280
- SetLogonRecovery, 280
- SetLogonTimeout, 280
- SetNumOfLstErrs, 280
- SetReserveConnHeadSet, 280
- SetStatFileName, 280
- settings:moagent32, 67
- SetUseDIIDbs, 281
- Start the agent binary, 20
- State, 16, 28
- states, 16, 25
- syntax, 66
- Avaya Proactive Contact:call completion codes;agent:call completion codes, 222
- Telnet, 4
- terminators, 25
- Terminology, 16
- Testing applications, 19
- Time, 287
- Transferring calls, 14
- types, 9, 11



## Avaya Proactive Contact Agent API 5.2

unit work list, 16, 45

UnitID, 287

VB, 54

Visual Basic, 54

Visual C++, 55

Volume, 287

WaitMsg, 287

Winsockerrors, 69

Work\_class, 68

Wrap up calls, 14