Notice
While reasonable efforts were made to ensure that the information in this document was complete and accurate at the time of printing, Avaya Inc. cannot assume no liability for any errors. Changes and corrections to the information in this document might be incorporated in future releases.

Documentation disclaimer
Avaya Inc. is not responsible for any modifications, additions, or deletions to the original published version of this documentation unless such modifications, additions, or deletions were performed by Avaya. Customer and/or End User agree 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 the Customer or End User.

Link disclaimer
Avaya Inc. is not responsible for the contents or reliability of any linked Web sites referenced elsewhere within this documentation, and Avaya does not necessarily endorse the products, services, or information described or offered within them. We cannot guarantee that these links will work all the time and we have no control over the availability of the linked pages.

Warranty
Avaya Inc. provides a limited warranty on this product. 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 through the Avaya Support Web site: http://www.avaya.com/support

License
USE OR INSTALLATION OF THE PRODUCT INDICATES THE END USER'S ACCEPTANCE OF THE TERMS SET FORTH HEREIN AND THE GENERAL LICENSE TERMS AVAILABLE ON THE AVAYA WEB SITE http://support.avaya.com/LicenseInfo/ (“GENERAL LICENSE TERMS”). IF YOU DO NOT WISH TO BE BOUND BY THESE TERMS, YOU MUST RETURN THE PRODUCT(S) TO THE POINT OF PURCHASE WITHIN TEN (10) DAYS OF DELIVERY FOR A REFUND OR CREDIT.

Avaya grants End User a license within the scope of the license types described below. 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 End User. *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. *Software* means the computer programs in object code, originally licensed by Avaya and ultimately utilized by End User, whether as stand-alone Products or pre-installed on Hardware. *Hardware* means the standard hardware Products, originally sold by Avaya and ultimately utilized by End User.

License type(s)

**Designated System(s) License (DS).** End User may install and use each copy of the Software on only one Designated Processor, unless a different number of Designated Processors is indicated in the Documentation or other materials available to End User. Avaya may require the Designated Processor(s) to be identified by type, serial number, feature key, 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 Product that permits one user to interface with the Software. Units may be linked to a specific, identified Server.

**Database License (DL).** Customer may install and use each copy of the Software on one Server or on multiple Servers provided that each of the Servers on which the Software is installed communicate with no more than a single instance of the same database.

**CPU License (CP).** End User may install and use each copy of the Software on a number of Servers up to the number indicated by Avaya provided that the performance capacity of the Server(s) does not exceed the performance capacity specified for the Software. End User may not re-install or operate the Software on Server(s) with a larger performance capacity without Avaya's prior consent and payment of an upgrade fee.

**Named User License (NU).** Customer may: (i) install and use the Software on a single Designated Processor or Server per authorized Named User (defined below); or (ii) install and use the Software on a Server so long as only authorized Named Users access and use the Software. *Named User,* means a user or device that has been expressly authorized by Avaya to access and use the Software. At Avaya's sole discretion, a "Named User" may be, without limitation, designated by name, corporate function (e.g., webmaster or helpdesk), an e-mail or voice mail account in the name of a person or corporate function, or a directory entry in the administrative database utilized by the Product that permits one user to interface with the Product.

**Shrinkwrap License (SR).** With respect to Software that contains elements provided by third party suppliers, End User may install and use the Software in accordance with the terms and conditions of the applicable license agreements, such as "shrinkwrap" or "clickwrap" license accompanying or applicable to the Software ("Shrinkwrap License"). The text of the Shrinkwrap License will be available from Avaya upon End User's request (see *Third-party Components* for more information).

Copyright
Except where expressly stated otherwise, the Product is protected by copyright and other laws regarding proprietary rights. Unauthorized reproduction, transfer, and or use can be a criminal, as well as a civil, offense under the applicable law.

Third-party components
Certain software programs or portions thereof included in the Product may contain software distributed under third party agreements ("Third Party Components"), which may contain terms that expand or limit rights to use certain portions of the Product ("Third Party Terms"). Information identifying Third Party Components and the Third Party Terms that apply to them is available on the Avaya Support Web site: http://support.avaya.com/ThirdPartyLicense/

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 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 Web site: http://www.avaya.com/support

Trademarks
Avaya and the Avaya logo are either registered trademarks or trademarks of Avaya Inc. in the United States of America and/or other jurisdictions. All other trademarks are the property of their respective owners.

Downloading documents
For the most current versions of documentation, see the Avaya Support Web site: http://www.avaya.com/support

Ordering documents

**Voice:** +1-207-866-6701
1-800-457-1235 (Toll-free, U.S. and Canada only)

**Fax:** +1-207-866-6095

**Write:** Globalware Solutions
200 Ward Hill Avenue
Haverhill, MA 01835 USA
Attention: Avaya Account Manager

**Web:** http://www.avaya.com/support

**E-mail:** totalware@gwsmail.com

**Order:** Document No. 07-601189
April 2006

Avaya support
Avaya provides a telephone number for you to use to report problems or to ask questions about your product. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Support Web site: http://www.avaya.com/support
Contents

Preface ................................................................. 7
Purpose .............................................................. 7
Audience ............................................................. 7
Reasons for reissue .................................................. 8
Related documents .................................................. 8

Chapter 1: Overview ................................................. 9
Avaya Proactive Contact system functions ....................... 10
  Multiple dialers .................................................. 10
  Pods ..................................................................... 10
  Calling lists ........................................................ 11
  Agent Blending ..................................................... 12
Agent tasks overview ................................................ 15
  Agent types ........................................................ 16
  Log in to Avaya Proactive Contact ......................... 17
  Join jobs ............................................................ 18
  Handle calls ....................................................... 18
  Wrap up calls ..................................................... 21
  Log out of Avaya Proactive Contact ...................... 21
Understanding the Avaya Proactive Contact Agent API .... 23
  Terminology ....................................................... 24
  Agent applications ............................................. 24
  Agent binary ..................................................... 27
  Agent API messages ........................................... 27
  Agent application initiated messages .................... 28
  Agent binary initiated messages .......................... 28
  Message formats ............................................... 28
  Agent states ..................................................... 32
Outbound job commands (by agent state) ..................... 35
Outbound job message scenarios ............................... 36
Moagent32.log ....................................................... 40
Sample agent application session ............................... 41
Sample managed dialing job ..................................... 47
Sample unit work list job ......................................... 52
Sample agent owned recall ....................................... 55

Chapter 2: Using Avaya Proactive Contact Agent API DLL .. 61
Developer requirements ............................................. 62
The Avaya Proactive Contact Agent API and COM ......... 63
Work with the Moagent32.dll ..................................... 64
## Contents

- Connect to the Moagent32.dll ................................................. 64
- Use the interface methods .................................................. 67
- Handle notification events from Avaya Proactive Contact .......... 69
- Set default properties ....................................................... 72
- Code samples ........................................................................ 72
- Use Moagent32.dll dialog boxes .......................................... 75
- The Moagent32 interface ..................................................... 78
- Method syntax ....................................................................... 79
- Moagent32.ini settings .......................................................... 80

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Logon]</td>
<td>80</td>
</tr>
<tr>
<td>[Work_class]</td>
<td>80</td>
</tr>
<tr>
<td>[Dialog_box_text]</td>
<td>81</td>
</tr>
<tr>
<td>[Agent_state]</td>
<td>81</td>
</tr>
<tr>
<td>[Server_return_codes]</td>
<td>81</td>
</tr>
<tr>
<td>[Winsockerrors]</td>
<td>81</td>
</tr>
</tbody>
</table>

Chapter 3: Commands and notification events .......................... 83
- AGTAdjustHeadset .................. 84
- AGTAORNotify ...................... 86
- AGTAttachJob ...................... 88
- AGTAutoReleaseLine .............. 90
- AGTAvalWork ....................... 91
- AGTCallNotify ..................... 94
- AGTClearDataSet .................. 98
- AGTConnHeadset ................... 100
- AGTDetachJob ...................... 102
- AGTDialDigit ...................... 104
- AGTDialerMode ..................... 106
- AGTDisconnHeadset ............... 108
- AGTDoNotCall ..................... 110
- AGTDumpData ...................... 111
- AGTEchoOff ........................ 116
- AGTEchoOn .......................... 118
- AGTFinishedItem .................. 119
- AGTFreeHeadset ................... 121
- AGTGetHeadsetVol ................. 123
- AGTGetScreen ..................... 125
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTHangupCall</td>
<td>127</td>
</tr>
<tr>
<td>AGTHeadsetConnBroken</td>
<td>129</td>
</tr>
<tr>
<td>AGTHoldCall</td>
<td>131</td>
</tr>
<tr>
<td>AGTHookFlashLine</td>
<td>133</td>
</tr>
<tr>
<td>AGTICbAbort</td>
<td>135</td>
</tr>
<tr>
<td>AGTICbFeNotif</td>
<td>137</td>
</tr>
<tr>
<td>AGTICbOffline</td>
<td>139</td>
</tr>
<tr>
<td>AGTICbOnline</td>
<td>141</td>
</tr>
<tr>
<td>AGTJobEnd</td>
<td>143</td>
</tr>
<tr>
<td>AGTJobTransLink</td>
<td>145</td>
</tr>
<tr>
<td>AGTJobTransRequest</td>
<td>147</td>
</tr>
<tr>
<td>AGTLListCallbackFmt</td>
<td>149</td>
</tr>
<tr>
<td>AGTLListCallFields</td>
<td>151</td>
</tr>
<tr>
<td>AGTLListCallLists</td>
<td>154</td>
</tr>
<tr>
<td>AGTLListDataFields</td>
<td>156</td>
</tr>
<tr>
<td>AGTLListJobs</td>
<td>159</td>
</tr>
<tr>
<td>AGTLListKeys</td>
<td>161</td>
</tr>
<tr>
<td>AGTLListScreens</td>
<td>163</td>
</tr>
<tr>
<td>AGTLListState</td>
<td>164</td>
</tr>
<tr>
<td>AGTLListUnits</td>
<td>166</td>
</tr>
<tr>
<td>AGTLogIoStart</td>
<td>168</td>
</tr>
<tr>
<td>AGTLogIoStop</td>
<td>174</td>
</tr>
<tr>
<td>AGTLogoff</td>
<td>176</td>
</tr>
<tr>
<td>AGTLogoffAcld</td>
<td>178</td>
</tr>
<tr>
<td>AGTLogon</td>
<td>180</td>
</tr>
<tr>
<td>AGTLogonAcld</td>
<td>182</td>
</tr>
<tr>
<td>AGTManagedCall</td>
<td>185</td>
</tr>
<tr>
<td>AGTManualCall</td>
<td>187</td>
</tr>
<tr>
<td>AGTMoFlashBlind</td>
<td>189</td>
</tr>
<tr>
<td>AGTMoFlashSupv</td>
<td>193</td>
</tr>
<tr>
<td>AGTNoFurtherWork</td>
<td>197</td>
</tr>
<tr>
<td>AGTPreviewRecord</td>
<td>199</td>
</tr>
<tr>
<td>AGTReadField</td>
<td>202</td>
</tr>
<tr>
<td>AGTReadyNextItem</td>
<td>205</td>
</tr>
<tr>
<td>AGTReceiveMessage</td>
<td>207</td>
</tr>
<tr>
<td>AGTReleaseLine</td>
<td>209</td>
</tr>
</tbody>
</table>
Preface

This section contains the following topics:

- **Purpose** on page 7
- **Audience** on page 7
- **Reasons for reissue** on page 8
- **Related documents** on page 8

---

Purpose

The purpose of this guide is to provide detailed information about Avaya Proactive Contact, formerly named Avaya Predictive Dialing System (PDS) version 12.0.

The Avaya Proactive Contact Agent API (Agent API) is a developer’s tool that allows you to produce agent applications for Avaya Proactive Contact. Agent API is a set of commands and event notifications that enable application developers to create a customized user interface for handling Avaya Proactive Contact agent operations.

---

Audience

This guide is for personnel who develop agent applications for Avaya Proactive Contact.

The Agent API consists of an Avaya Proactive Contact server component and a PC component. The server component, **agent binary**, is an Avaya Proactive Contact server-side process that links to the Avaya Proactive Contact agent process and manages agent socket connections. The PC component, **agent DLL**, is the COM server through which client applications interact with the agent binary process.

Before beginning development, you need to be familiar with the following:

- Avaya Proactive Contact
- Avaya Proactive Contact Agent API
- Your agent application requirements
- Your development tool, such as Visual Basic, PowerBuilder, or Visual C++
Reasons for reissue

This issue of the API Reference includes information about the DDE interface and updates for the Avaya Proactive Contact with CTI system deployment option.

Note:
You can 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.

Related documents

The Avaya Proactive Contact documentation set consists of:

- What’s New in Avaya Proactive Contact
- Administering Avaya Proactive Contact
- Using Supervisor
- Using Analyst
- Using Agent
- Avaya Proactive Contact Software Reference - Scripts
- Avaya Proactive Contact Software Reference - Configuration Files
- Avaya Proactive Contact Software Reference - Binaries
- Avaya Proactive Contact Software Developer’s Kit (SDK)
- Avaya Proactive Contact 3.0 Vlterm Reference
- Planning and Prerequisites for Avaya Proactive Contact
- Configuring and Installing Avaya Proactive Contact
- Avaya Proactive Contact Safety and Regulatory Information
Chapter 1: Overview

The Avaya Proactive Contact 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](#) on page 10
- [Agent tasks overview](#) on page 15
- [Understanding the Avaya Proactive Contact Agent API](#) on page 23
# 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 Avaya Proactive Contact 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 Avaya Proactive Contact system can include multiple dialers. You can connect up to four dialers through a middle-tier structure.

An Avaya Proactive Contact system that is connected through a middle-tier structure is a pod.

Your Avaya Proactive Contact 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. Avaya Proactive Contact 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.

Avaya 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, Avaya Proactive Contact 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
Overview

After calling activities and at a scheduled time, Avaya Proactive Contact 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 Avaya Proactive Contact 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 Avaya Proactive Contact 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.

Avaya Proactive Contact 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.

Avaya Proactive Contact 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.
- The dialer 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.
Overview

Note:
For each OB_ONLY domain group, you configure the number of queued calls before agents release to inbound.
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 Avaya Proactive Contact Agent software application.

This section contains the following topics:

- **Agent types** on page 16
- **Log in to Avaya Proactive Contact** on page 17
- **Join jobs** on page 18
- **Handle calls** on page 18
- **Wrap up calls** on page 21
- **Log out of Avaya Proactive Contact** on page 21

The following diagram illustrates the discussion. Both focus on the activities agents perform without an agent application. [Understanding the Avaya Proactive Contact Agent API](#) on page 23 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.
Log in to Avaya 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 Avaya Proactive Contact.

   **Note:**
   On Avaya 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 Avaya Proactive Contact how to make the audio connection to the agent through the agent headset or workstation ID.

3. Avaya Proactive Contact confirms that the ID is valid and then displays the agent interface. The default interface is the Avaya Proactive Contact Agent.

4. Avaya Proactive Contact establishes a voice connection to the agent headset. The agent hears a welcome message when Avaya Proactive Contact makes the connection.
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, Avaya Proactive Contact will not accept the job selection or the agent may not 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 Avaya Proactive Contact that they are available to handle calls.

Handle calls

Avaya Proactive Contact distinguishes between calls answered by a person and calls answered by an electronic device. Avaya 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 Avaya 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 may 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 Avaya 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.
- Avaya 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

Avaya 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. Avaya 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 Avaya 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.

Avaya 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 Avaya 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, Avaya 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 Avaya 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 Avaya Proactive Contact with Intelligent Call Blending to handle inbound calls. After the agent joins a job, Avaya 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 Avaya 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 Avaya Proactive Contact.

**Blend calling**

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

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

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 Avaya Proactive Contact blend agents. When there are no inbound calls overflowing from the ACD, Avaya Proactive Contact connects outbound calls to blend agents.

On systems without an ACD, Avaya 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 Avaya 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 Avaya Proactive Contact to place the call. The system calls the phone number.

Calling acquired phone numbers

The agent may tell Avaya 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 Avaya 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 Avaya 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, Avaya Proactive Contact postpones the call.

---

**Wrap up calls**

When the conversation ends, the agent may be finished with the customer’s record and ready for another call, or the agent may 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 Avaya 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. Avaya 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. Avaya Proactive Contact saves the record update and releases the record. It then records that the agent is available for another call.

---

**Log out of Avaya Proactive Contact**

Logging out has three steps: leaving a job, disconnecting the agent’s headset, and logging out of Avaya 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
Overview

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

Avaya 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, Avaya Proactive Contact disconnects from the agent’s headset and clears the memory buffer reserved for that headset identification. On most Avaya Proactive Contact agent interfaces, the logout process automatically disconnects the headset.

Log out

On the Avaya Proactive Contact UNIX-based Agent Menu, the agent selects logout. Avaya Proactive Contact logs the agent out of the system and returns the workstation to the Avaya 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 Avaya Proactive Contact.
- Note that if the workstation also connects to a host system, the agent may need to log out of the host system after logging out of Avaya Proactive Contact.

On the Avaya 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 Avaya Proactive Contact to terminate the calling session by disconnecting any call in progress and immediately logging the agent out of the system.
Understanding the Avaya Proactive Contact Agent API

The Avaya Proactive Contact Agent API (Agent API) lets you build a graphical user interface (GUI) for Avaya 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 Avaya Proactive Contact server that enables an API for custom applications.

The agent application can run on any workstation that can establish a standard TCP/IP socket connection to Avaya Proactive Contact.

This section contains the following topics:

- Terminology on page 24
- Agent applications on page 24
- Agent binary on page 27
- Agent API messages on page 27
- Agent application initiated messages on page 28
- Agent binary initiated messages on page 28
- Message formats on page 28
- Agent states on page 32
- Outbound job commands (by agent state) on page 35
- Outbound job message scenarios on page 36
- Moagent32.log on page 40
- Sample agent application session on page 41
- Sample managed dialing job on page 47
- Sample unit work list job on page 52
- Sample agent owned recall on page 55
Overview

Terminology

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

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent application</td>
<td>The application Avaya Proactive Contact agents use to work with customers. The agent application runs on an agent's workstation.</td>
</tr>
<tr>
<td>Agent binary</td>
<td>The agent process runs on Avaya Proactive Contact and provides command driven access. Some Avaya Proactive Contact documentation may refer to the agent process as the Avaya Proactive Contact process.</td>
</tr>
<tr>
<td>Avaya Proactive Contact server</td>
<td>The network server running the Avaya Proactive Contact application.</td>
</tr>
</tbody>
</table>

Agent applications

The agent application is the application agents logged in to Avaya 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. For help understanding typical agent tasks, see Agent tasks overview on page 15.

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 Avaya Proactive Contact.
Understanding the Avaya Proactive Contact Agent API

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 Avaya 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 Avaya Proactive Contact

An agent application developed with the Agent API can check ACD status and provide separate logout processes for ACD agents and Avaya 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 may need to log out of the host system connection after logging out of Avaya Proactive Contact. The Agent API does not provide for logging out of a host system connection.

Exit Avaya Proactive Contact

Agent API does not support agent-initiated emergency exits from Avaya 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 Avaya Proactive Contact to test your agent application. We recommend that you work closely with the Avaya 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 Avaya Proactive Contact, Avaya Proactive Contact test systems are available that simulate call center activity.
Important:
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 Avaya 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 Avaya Proactive Contact to start the agent process when Avaya Proactive Contact starts. The agent application connects to Avaya Proactive Contact through a TCP/IP connection. 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 Avaya 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 Avaya Proactive Contact, open a command line interface to Avaya 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 (Avaya 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 Avaya Proactive Contact.
Data messages (D)

Data messages consist of an Avaya Proactive Contact message and the requested data. The Avaya 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 Avaya 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:

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

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.

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyword</td>
<td>A unique identifier for agent commands and events.</td>
<td>20 bytes</td>
</tr>
<tr>
<td>Type</td>
<td>Command or event message type.</td>
<td>1 byte</td>
</tr>
<tr>
<td></td>
<td>C - Command from agent application to agent binary</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P - Pending response from agent binary to agent application</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D - Data message from agent binary to agent application</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R - Response from agent binary to agent application</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B - Busy message from agent binary to agent application (very rare)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N - Notification event from agent binary to agent application</td>
<td></td>
</tr>
<tr>
<td>Client</td>
<td>Reserved for future use. “Client” appears in the message header.</td>
<td>20 bytes</td>
</tr>
<tr>
<td>Process ID</td>
<td>Reserved for future use. “ProcID” appears in the message header.</td>
<td>6 bytes</td>
</tr>
<tr>
<td>Invoke ID</td>
<td>Reserved for future use. “InvokeID” appears in the message header.</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Number of segments</td>
<td>Number of data segments in the data portion of the record.</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Type</th>
<th>Client</th>
<th>Process ID</th>
<th>Invoke ID</th>
<th>Number of segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogon</td>
<td>C</td>
<td>Client</td>
<td>ProclID</td>
<td>Invoke ID</td>
<td>2</td>
</tr>
<tr>
<td>20 bytes</td>
<td>1 byte</td>
<td>20 bytes</td>
<td>6 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

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 l (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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>All</td>
</tr>
<tr>
<td>B</td>
<td>Blend</td>
</tr>
<tr>
<td>O</td>
<td>Outbound</td>
</tr>
<tr>
<td>I</td>
<td>Inbound</td>
</tr>
<tr>
<td>M</td>
<td>Managed dialing</td>
</tr>
</tbody>
</table>

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

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

Avaya Proactive Contact message codes are one alpha character followed by five numbers. The alpha character at the beginning of the Avaya Proactive Contact message code indicates the message type.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Message</td>
</tr>
<tr>
<td>E</td>
<td>Error</td>
</tr>
<tr>
<td>S</td>
<td>Status</td>
</tr>
</tbody>
</table>

There are two acceptable status types.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Inactive</td>
</tr>
<tr>
<td>A</td>
<td>Active</td>
</tr>
</tbody>
</table>

(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.

<table>
<thead>
<tr>
<th>Separator</th>
<th>Status</th>
<th>Separator</th>
<th>Message number</th>
<th>Separator</th>
<th>Data</th>
<th>Message terminator</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>(0 or 1)</td>
<td>1 byte</td>
<td>6 bytes</td>
<td>1 byte</td>
<td>Variable length</td>
<td>1 byte</td>
</tr>
</tbody>
</table>

**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.
Overview

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.

<table>
<thead>
<tr>
<th>Separator</th>
<th>Data</th>
<th>Message terminator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>1 byte [x1E]</td>
<td>1 byte [x03] or [x17]</td>
</tr>
<tr>
<td>Example</td>
<td>§ 0 § AREA2 § 206</td>
<td>¶</td>
</tr>
</tbody>
</table>

Data from agent binary to agent application

Note:

For more information about message numbers and the DLL, see Chapter 2: Using Avaya Proactive Contact Agent API DLL on page 61.

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 Avaya Proactive Contact. Agent states can include:

- Not logged in to Avaya Proactive Contact
- Logged in to Avaya 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.
Overview

Managed Job

Socket connect to the dialer

Log in to the system

DetachJob ()

Attached to a job in idle mode

Join a job

Logon (Uid,PW)

Logged in

AttachJob (jobname)

Available for work, not taking calls yet

NoFurtherWork ()

Waiting for a call

AvailWork ()

ReadyNextItem ()

NoFurtherWork ()

Waiting for a call

ManageCall ()

Issued by client or system after timeout

Manager record open

Dialer Event

F

FinishItem ()

Wrap up calls

G

Pending Call completion Close record

Log out of the system

H

Detach from the job

Logged out

Note:
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 describes the outbound job commands by agent state:

<table>
<thead>
<tr>
<th>Commands Available</th>
<th>Agent State(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>State “A” commands</td>
<td>Logon</td>
</tr>
<tr>
<td>State “B” commands</td>
<td>SENDMESSAGE</td>
</tr>
<tr>
<td></td>
<td>LogonAcd (Agent Blending systems only, using Predictive Agent Blending or Proactive Agent Blending)</td>
</tr>
<tr>
<td></td>
<td>LogloStart, LogloStop</td>
</tr>
<tr>
<td></td>
<td>ReserveHeadset, ConnHeadset</td>
</tr>
<tr>
<td></td>
<td>EchoOn, EchoOff (-t option only)</td>
</tr>
<tr>
<td></td>
<td>AttachJob</td>
</tr>
<tr>
<td></td>
<td>DumpData, SetWorkClass</td>
</tr>
<tr>
<td></td>
<td>ListCallFields, ListCallLists, ListState, ListJobs</td>
</tr>
<tr>
<td></td>
<td>Logoff</td>
</tr>
<tr>
<td>State “C” commands</td>
<td>AvailWork</td>
</tr>
<tr>
<td></td>
<td>ClearDataSet</td>
</tr>
<tr>
<td></td>
<td>DisconnHeadset, FreeHeadset</td>
</tr>
<tr>
<td></td>
<td>ListCallbackFmt, ListDataFields, ListKeys, ListUnits</td>
</tr>
<tr>
<td></td>
<td>SetDataField, SetNotifyKeyFields, SetUnit, SetWorkClass</td>
</tr>
<tr>
<td></td>
<td>DetachJob</td>
</tr>
<tr>
<td>State “D” commands</td>
<td>ReadyNextItem</td>
</tr>
<tr>
<td>State “E” commands</td>
<td>ManagedCall, PreviewRecord (Managed Dialing job only)</td>
</tr>
<tr>
<td></td>
<td>UpdateField, ReadField</td>
</tr>
<tr>
<td></td>
<td>MoFlashSupv, MoFlashBlind, DialDigit, TransferCall</td>
</tr>
<tr>
<td></td>
<td>HangupCall, HoldCall, ManualCall, HookFlashLine</td>
</tr>
<tr>
<td></td>
<td>DoNotCall</td>
</tr>
<tr>
<td></td>
<td>FinishedItem, NoFurtherWork, ReleaseLine</td>
</tr>
<tr>
<td></td>
<td>SetCallBack</td>
</tr>
<tr>
<td></td>
<td>AdjustHeadset, GetHeadSetVol</td>
</tr>
<tr>
<td>State “F” events from Avaya Proactive Contact</td>
<td>ReceiveMessage, SystemError, CallNotify</td>
</tr>
<tr>
<td></td>
<td>HeadsetConnBroken, JobEnd</td>
</tr>
<tr>
<td></td>
<td>TiicbAbort, TiicbFeNotify, TiicbOffline, TiicbOnline</td>
</tr>
<tr>
<td></td>
<td>JobTransLink, JobTransRequest</td>
</tr>
<tr>
<td>State “G” commands</td>
<td>DetachJob</td>
</tr>
<tr>
<td>State “H” commands</td>
<td>Logoff</td>
</tr>
<tr>
<td></td>
<td>LogoffAcd (Agent Blending systems only, using Predictive Agent Blending or Proactive Agent Blending)</td>
</tr>
</tbody>
</table>
Outbound job message scenarios

The following diagrams show sample message scenarios during outbound calling when an agent requests to log out of Avaya 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 an Avaya Proactive Contact Intelligent Call Blending system.
Understanding the Avaya Proactive Contact Agent API

Additional Agents Remain on the Outbound Job

Agent State

Agent on call

D
FinishedItem (FI)

ReadNextItem (RNI)

ReadNextItem (RNI)

On next call

E
FinishedItem (FI)

NoFurtherWork (NFW)

Attached job (idle)

C
Attached job (idle)

G
DetachJob (DJ)

H
Logoff (LO)

Logged out

Client

Server

C FinishedItem

R FinishedItem

C ReadyNextItem

R ReadyNextItem

N CallNotify

N CallNotify

N CallNotify

C NoFurtherWork

R NoFurtherWork

C DetachJob

R DetachJob

C LogOff

R LogOff

*If the system completes the calling list, it sends the ACTIVATION notification to the agent application. The agent would join another job or log out of the system.
Overview

Last Outbound Agent on the Job Requests to Log Out

Agent State

**Agent on a call**

- **D**
  - FinishedItem (FI)
  - RequiresAction (RNI)
  - On next call

**E** NoFurtherWork (NFW)

**Still on a call**

- **E**
  - ReleaseLine (RL)
- **F**
  - FinishedItem (FI)

**D**

- RequiresAction (RNI)
  - On a queue call

- FinishedItem (FI)

1 2 3

**Client**

- C FinishedItem
- R FinishedItem
- C RequiresAction
- R RequiresAction
- N CallNotify

**Server**

- C NoFurtherWork
- R NoFurtherWork
- C ReleaseLine
- R ReleaseLine
- C RequiresAction
- R RequiresAction
- N CallNotify
- N CallNotify
- N CallNotify
- C FinishedItem
- R FinishedItem

*If RNI returns an error, go to 1
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 may look like the following:

Client > 11:35:55 AGTLogon C102 203 304 2
04
****

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 Chapter 2: Using Avaya Proactive Contact Agent API DLL on page 61 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 Avaya Proactive Contact, selects a job, performs job setup activities, joins a job, and handles calls. When finished working, the agent logs off Avaya 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
<quilcene - your Avaya Proactive Contact server name>
agent
04
****
1
Client < 11:35:55 AGTSTART NAgent server 1539 0 2
0
AGENT_STARTUP
2
Client > 11:35:55 AGTLogon C102 203 304 2
04
****
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
4
6
Client < 11:35:55 AGTReserveHeadset PAgent server 1539 304 2
0
Overview

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
O
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 AGTNotifyKeyField C100 200 300 2
0 ACCTNUM
19 Client < 11:39:50 AGTNotifyKeyField RAgent server 1539 300 2
0 M00000
20 Client > 11:40:19 AGTSetDataField C100 200 300 2
O BAL
21 Client < 11:40:19 AGTSetDataField RAgent server 1539 300 2
0 M00000
22 Client > 11:40:30 AGTSetDataField C100 200 300 2
O CREDLIN
Understanding the Avaya Proactive Contact Agent API

Line | Meaning
--- | ---
1 | The agent application connection to Avaya 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 Avaya 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 Avaya Proactive Contact. This number represents a key code or an extension, depending on Avaya Proactive Contact and the type of headset connection.
11-13 | The agent application gets a list of all jobs on Avaya 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 Avaya 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

5-10 | The agent application reserves and connects to a numbered headset on Avaya Proactive Contact. This number represents a key code or an extension, depending on Avaya Proactive Contact and the type of headset connection.

26 Client < 11:41:27 AGTAvailWork RAgent server 1539 300 2

27 Client > 11:41:35 AGTRedyNextItem C100 200 300 0
28 Client < 11:41:36 AGTRedyNextItem RAgent server 1539 300 2

29 Client < 11:41:56 AGTCallNotify NAgent server 1539 0 5

30 Client < 11:41:56 AGTCallNotify NAgent server 1539 0 4

31 Client < 11:41:56 AGTCallNotify NAgent server 1539 0 2
Overview

32 Client > 11:43:00 AGTReadField C100 200 300 2
   PHONE2
33 Client < 11:43:01 AGTReadField DAgent server 1539 300 3
   0
   M00001
   PHONE2,N,10,0000000000
34 Client < 11:43:01 AGTReadField RAgent server 1539 300 2
   0
   M00000
35 Client > 11:43:18 AGTReadField C100 200 300 2
   O
   AREA2
36 Client < 11:43:18 AGTReadField DAgent server 1539 300 3
   0
   M00001
   AREA2,N,3,000
37 Client < 11:43:18 AGTReadField RAgent server 1539 300 2
   0
   M00000

<table>
<thead>
<tr>
<th>Line</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>24-26</td>
<td>The agent joins the job. The agent application logs in to the job with the AGTAvailWork command.</td>
</tr>
<tr>
<td>27-28</td>
<td>The agent application tells Avaya Proactive Contact the agent is ready to receive a call.</td>
</tr>
<tr>
<td>29-31</td>
<td>Avaya 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 Avaya Proactive Contact connects the agent to the client’s call.</td>
</tr>
<tr>
<td>32-37</td>
<td>The agent requests additional information from the customer record (the client’s second phone number and area code). Avaya Proactive Contact returns the information, and the agent application displays it.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Line</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>38-41</td>
<td>The agent updates the second phone number and area code. The agent application sends the new information to Avaya Proactive Contact. (To request and update information from the customer record, the agent application must know the calling list fields.)</td>
</tr>
<tr>
<td>42-51</td>
<td>The agent tells Avaya 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.</td>
</tr>
</tbody>
</table>
Overview

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
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 Avaya 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). Avaya 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 Avaya Proactive Contact know the agent is ready for
        another call.

59-61  Avaya 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
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 Avaya 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 2
04
****
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 < 12:08:49 AGTConnHeadset RAgent server 1578 304 2

Line Meaning

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 Avaya Proactive Contact but cannot take customer calls.

69-70 The agent logs off Avaya Proactive Contact.
Overview

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

| 18 Client > 12:12:14 AGTListDataFields C100 200 300 1     |
| 19 Client < 12:12:14 AGTListDataFields DAgent server 1578 300 50 |

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
Understanding the Avaya Proactive Contact Agent API

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  
CODE, 2, C, F  
ENTRYDATE, 10, D, F  
STATUSFLAG, 1, C, F  
RECALLDATE, 10, D, F  
RECALLTIME, 8, T, F  
DAYSCTN, 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, N, F  
RECALLPHONE, 2, C, F

<table>
<thead>
<tr>
<th>Line</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>18-20</td>
<td>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.</td>
</tr>
</tbody>
</table>

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  
DUPERC, 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
Overview

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
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
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

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 Avaya Proactive Contact the agent is ready to receive the first preview record.
Understanding the Avaya Proactive Contact Agent API

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 Avaya Proactive Contact that the agent can preview the record.
33-36   The Avaya 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 Avaya 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
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 Avaya Proactive Contact.

Client > 11:05:18 Connection Parameters
<quilcene - 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 2
0
****
3 Client < 11:05:19 AGTLogon PAgent server 1577 304 2
0

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 Avaya Proactive Contact.

43-44  The agent application tells Avaya Proactive Contact the agent is ready to receive the next preview record.

45-47  The agent application sends an AGTPreviewRecord command to tell Avaya 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 Avaya Proactive Contact.
Understanding the Avaya Proactive Contact Agent API

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
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
O
ACCTNUM
17 Client < 11:06:05 AGTSetNotifyKeyFieldRAgent server 1577 300 2
0
M00000
18 Client > 11:06:17 AGTSetDataField C100 200 300 2
O
BAL
19 Client < 11:06:17 AGTSetDataField RAgent server 1577 300 2

April 2006 53
### Overview

<table>
<thead>
<tr>
<th>Line</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The agent application connection to Avaya 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.</td>
</tr>
<tr>
<td>2-4</td>
<td>The agent enters a user name and password. Avaya Proactive Contact accepts the agent log in. When commands require additional processing by Avaya Proactive Contact, the agent binary issues a message with a code of S28833. This indicates the command is pending.</td>
</tr>
<tr>
<td>5-10</td>
<td>The agent application reserves the headset ID and connects to headset number 4 on Avaya Proactive Contact. This number could represent a key code or an extension, depending on Avaya Proactive Contact and the type of headset connection.</td>
</tr>
<tr>
<td>11-13</td>
<td>The agent application gets a list of all jobs on Avaya 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.</td>
</tr>
<tr>
<td>14-15</td>
<td>The agent selects the Unit Work List job. The agent application attaches the job.</td>
</tr>
<tr>
<td>16-17</td>
<td>The agent application selects the customer record key field for Avaya Proactive Contact to send when connecting a call to the agent.</td>
</tr>
<tr>
<td>18-21</td>
<td>The agent application selects two additional fields for Avaya Proactive Contact to send with call notifications.</td>
</tr>
</tbody>
</table>

22 Client > 11:06:43 AGTListUnits C100 200 300 0
23 Client < 11:06:43 AGTListUnits DAgent server 1577 300 3
0 M00000
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
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.

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.
Overview

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
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 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  Avaya 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.
The agent releases the record with a completion code 98 indicating it is an Agent Owned Recall.

The agent is ready to continue receiving calls on this job.

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.

The agent continues on the outbnd2 job.
About 10 minutes later, the agent application is notified that the agent must transfer to the outbnd1 job.

The agent application transfers the agent to outbnd1. Notice that the job setup notify key field and set data field are omitted.

The agent is now ready to start receiving calls on this job.

The agent application is notified of an Agent Owned Recall and the agent is sent the customer record and the outbound call.

The agent is finished with the customer record.
<table>
<thead>
<tr>
<th>Line</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>38-41</td>
<td>The agent is finished with the customer record.</td>
</tr>
<tr>
<td>42-43</td>
<td>The agent application is notified that the agent is now ready to go back to the outbnd2 job.</td>
</tr>
<tr>
<td>44-47</td>
<td>The agent application detaches from the outbnd1 job and attaches to the outbnd2 job.</td>
</tr>
<tr>
<td>48-52</td>
<td>The agent is now ready to start receiving calls again on outbnd2.</td>
</tr>
</tbody>
</table>
Chapter 2: Using Avaya Proactive Contact Agent API DLL

The Avaya 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 Avaya Proactive Contact network-connected agent workstation configuration.

This section contains the following topics:

- [Developer requirements](#) on page 62
- [The Avaya Proactive Contact Agent API and COM](#) on page 63
- [Work with the Moagent32.dll](#) on page 64
- [Use the interface methods](#) on page 67
- [Handle notification events from Avaya Proactive Contact](#) on page 69
- [Set default properties](#) on page 72
- [Use Moagent32.dll dialog boxes](#) on page 75
- [The Moagent32 interface](#) on page 78
- [Method syntax](#) on page 79
- [Moagent32.ini settings](#) on page 80
Developer requirements

The following are skills required of developers.

- Windows platform programming skills.
- Knowledge of Avaya Proactive Contact operation, interfaces, and methods.
- Knowledge of third-party development tools.
- Knowledge of COM (Component Object Model) architecture.
The Avaya 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 Avaya 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 Avaya Proactive Contact by specifying an object name defined within the client, associating the object with the Avaya 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:

```plaintext
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 Avaya Proactive Contact. Code samples for Visual Basic, PowerBuilder, and Visual C++ are provided.

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.

Note:
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.

![Diagram showing the connection process between client and Moagent32.dll](image)

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

Note:
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

// 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’.
  {
Using Avaya Proactive Contact Agent API DLL

```c
{ 
    if(SUCCEEDED(GHr = GptrServerStartup.CreateInstance(CLSID_IServerStartup)))
    {
        GptrMoagent = GptrServerStartup->PropIMoagent;

        //IMoagent interface. All methods (Avaya 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

Use the interface methods
The methods available within the IMoagent interface issue all Avaya Proactive Contact API
commands. For example, commands used to log in to Avaya Proactive Contact or transfer a
caller. These commands have the same name as the name of the method. A method is
available for each Avaya Proactive Contact API command. All input arguments are strings and
each method returns a Boolean type only when Avaya Proactive Contact completes the issued
command. That is, all Moagent32 methods are synchronous.
To issue an Avaya Proactive Contact command from your client application, you have to call a
method.
See The Moagent32 interface on page 78 the section later in this chapter for sample calls. See
Appendix D: Moagent32.dll interfaces on page 301 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)

April 2006

67


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_BUT_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 Avaya Proactive Contact

Avaya Proactive Contact sends messages, “notifications”, to your client application in response to events that occur on Avaya Proactive Contact. Notifications can either be solicited by your client application or sent unsolicited by Avaya Proactive Contact. For example, if your client application issues the command associated with the ReadyNextItem method, Avaya Proactive Contact returns the CallNotify notification which provides the customer data. HeadsetConnBroken is an example of an unsolicited notify sent by Avaya 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
- JobTransLink
- JobTransRequest
- PreviewRecord
- ReceiveMessage
- Start
- SystemError
- XferCustHangup
- XferTrunkHanup

For a description of each notification event, see Chapter 3: Commands and notification events on page 83. Some notification events are specific to the Moagent32.dll; data is not sent from Avaya Proactive Contact. See Appendix D: Moagent32.dll interfaces on page 301 for descriptions of these events.

The following code samples demonstrate how you would handle Avaya 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)
Using Avaya Proactive Contact Agent API DLL

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 Avaya 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 Avaya 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.
// This template is tied to the SINK_MAP macro
// (UniqueId - user choice, this class, Id for __IMOagent, Type Library, // version, version indice)
public IDispEventImpl<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()
Handle notification events from Avaya Proactive Contact

void __stdcall 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
}
Using Avaya Proactive Contact Agent API DLL

---

Set default properties

The default properties for the Moagent32.dll (IConfigure Interface) are listed in Appendix D: Moagent32.dll interfaces on page 301. You can specify a different property value from your client application.

Note:
The default SetCreateLogFile property is True. You may want to 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 Avaya 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.SetNumOfLateErrs = 5 'Set the number of errors to track during a client session. Use the method DllConfig.DisplayLastErrors to view a errors 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 Avaya 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

---

72 Avaya Proactive Contact Agent API Reference
// available to the client.
MosConfig.SetCreateLogFile = True // Create log file of all Avaya Proactive Contact-
//Moagent32.dll transactions.
MosConfig.SetLogFileName = "c:\moslog\moagent32.log"
MosConfig.SetCreateErrFile = False // Write all errors (AvayaMosaix, DLL,
// client) to log file.
MosConfig.SetErrMsgFile = "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 Avaya 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 _IConfig 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);

// make several standard dialog
//boxes available to the client.
m_pConfig->PutSetUseDllDbs = 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.DisplayLastError 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 Avaya 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 SetUseDllDbs need to be set to **True** (default value) to use these dialog boxes.

If you choose not to use the dialog boxes, set SetUseDllDbs 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)

![Adjust Headset dialog box](image1)

AttachJob ("",ErrCode,ErrText)

![Available jobs for agent type](image2)
Using Avaya Proactive Contact Agent API DLL

ManualCall ("",ErrCode,ErrText)

TransferCall ("",ErrCode,ErrText)

DialDigit ("",ErrCode,ErrText)
FinishedItem ("",ErrCode,ErrText)

SetCallback ("",ErrCode,ErrText)

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

SetWorkClass ("",ErrCode,ErrText)
The Moagent32 interface

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

- Methods with no Avaya Proactive Contact arguments that do not return data
- Methods with no Avaya 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 Avaya Proactive Contact API commands that have the same name as the name of the method. See Appendix D: Moagent32.dll interfaces on page 301 for a list of methods and their arguments.

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

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

**Note:**

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)

**Note:**

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 Avaya 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servername</td>
<td>Avaya Proactive Contact name</td>
</tr>
<tr>
<td>Portnumber</td>
<td>Avaya Proactive Contact port number. Set to 22700. Use this setting. If changed, the DLL will not connect to Avaya Proactive Contact.</td>
</tr>
<tr>
<td>Headset ID</td>
<td>Optional. If not included, Moagent32.dll will prompt for an ID during the login process.</td>
</tr>
</tbody>
</table>

[Work_class]

Contains all available agent type codes. While agents can use any of these codes, individual Avaya Proactive Contact installations may 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 Avaya Proactive Contact. It may, 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 Avaya Proactive Contact error codes that may return in response to Agent API commands. The error codes are part of the Avaya Proactive Contact binary files and cannot be changed.

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

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

[Winsockerrors]

Contains the text for winsock.ocx errors. Do not change message numbers.
Chapter 3: Commands and notification events

This section contains the commands and notification events that make up the Avaya Proactive Contact Agent API (Agent API). They are the methods used for IMoagent. See Appendix D: Moagent32.dll interfaces on page 301 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.

Note:
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 Avaya Proactive Contact system ID. For general information regarding command and event types, see Understanding the Avaya Proactive Contact Agent API on page 23.
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 Avaya Proactive Contact systems using OLIC cards.

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

Note:

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

Data parameters

<table>
<thead>
<tr>
<th>EarMouth</th>
<th>The setting to change: E (ear) or M (mouth).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume</td>
<td>The numeric (1-8) value for the volume setting. Use whole numbers only, no decimal values.</td>
</tr>
</tbody>
</table>
Return values

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

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

See also

- AGTGetHeadsetVol on page 123
- AGTHangupCall on page 127
- AGTManualCall on page 187
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 <ProcId> <InvokeID> 6 RS 0 RS M00001 RS CustomerName RS XferJobName RS UnitId RS OrigJobName

Where RS is the record separator.

Data parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomerName</td>
<td>The name of the customer that is being recalled.</td>
</tr>
<tr>
<td>XferJobName</td>
<td>The name of the job the agent is being transferred to for the recall.</td>
</tr>
<tr>
<td>UnitId</td>
<td>The unit work list value (if applicable) the agent specified for the original call.</td>
</tr>
<tr>
<td>OrigJobName</td>
<td>The name of the job where the recall was specified.</td>
</tr>
</tbody>
</table>

Return values

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

**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 Avaya 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§
§ = Message data separator, ASCII x1E (socket) or comma (tty)
<brJobName>‡
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameter

<table>
<thead>
<tr>
<th>JobName</th>
<th>The value is the name of an active job on Avaya Proactive Contact. Use up to 19 characters, 7-bit USASCII. The &lt;JobName&gt; is case sensitive and may not include special characters or embedded spaces.</th>
</tr>
</thead>
</table>
AGTAttachJob

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28804</td>
<td>Job &lt;JobName&gt; is not running (not active). This message usually indicates that a job is still starting.</td>
</tr>
<tr>
<td>E28889</td>
<td>The agent application is already attached to a job. The agent must clear the current job selection before selecting a new one.</td>
</tr>
<tr>
<td>E28890</td>
<td>Cannot open the job’s resource file. This message usually indicates an Avaya Proactive Contact system problem or that a job is still starting.</td>
</tr>
<tr>
<td>E29206</td>
<td>Cannot attach the &lt;Unit&gt; segment of shared memory that Avaya Proactive Contact is using for the job. Most jobs use the same unit of shared memory, defined in a configuration file on the Avaya Proactive Contact system. This message indicates a problem with the Avaya Proactive Contact system configuration.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S28833</td>
<td>Pending.</td>
</tr>
</tbody>
</table>

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](#) on page 91
- [AGTDetachJob](#) on page 102
- [AGTListJobs](#) on page 159
- [AGTLogon](#) on page 180
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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Complete</td>
</tr>
</tbody>
</table>

See also

- AGTXferCustHangup on page 244
- AGTXferTrunkHangup on page 245
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 AGTReadyNextItem when the agent is ready to receive a call from Avaya Proactive Contact.
To log the agent out of an active job, the agent application executes the AGTNoFurtherWork command.
Avaya Proactive Contact or the supervisor may also log the agent out of the job (see AGTJobEnd on page 143, AGTJobTransLink on page 145, AGTJobTransRequest on page 147).

Availability

Execute this command any time after AGTConnHeadset and AGTAttachJob for non-ACD agents.
Execute this command every time after AGTlcbOnline for ACD agents.

Format

AGTAvailWork C <Client> <ProcID> <InvokeID> 0‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None.
Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 may join the job. Job parameters include a maximum number of agents of each type (work class) that may log in to the job. See AGTSetWorkClass on page 231. |
| E28814 | Managed agents may not 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 may 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 may 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 may 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 may 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 may have become inactive since it was attached. Execute AGTListJobs to check the job status.      |
| E28899 | There is no available for work request pending. AGTAvailWork did not execute. This message may indicate an Avaya Proactive Contact system problem. |
E28900  § <message>‡
An Avaya 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 Avaya Proactive Contact has not acquired the agent for outbound calling.

E29000  Only Managed agents may join this job. The agent application agent type (work class) does not match the job type. Reset the agent type to M and retry.

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](#) on page 88
- [AGTConnHeadset](#) on page 100
- [AGTJobEnd](#) on page 143
- [AGTJobTransLink](#) on page 145
- [AGTJobTransRequest](#) on page 147
- [AGTListJobs](#) on page 159
- [AGTNoFurtherWork](#) on page 197
- [AGTSetWorkClass](#) on page 231
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 may 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 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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpMesg</td>
<td>OpMesg Message from call notification containing field information from the customer record configured in alljobs.dat on the Avaya Proactive Contact system. The default field is NAME. If the call is a voice and data transfer from another agent, the &lt;OpMesg&gt; is “TRANSFER CALL.” 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. The &lt;OpMesg&gt; for the pending notification is “AgtOwned Rc: &lt;Agent ID&gt; J: &lt;job name&gt; U: &lt;user ID&gt; T: HH.MM.SS.” The &lt;OpMesg&gt; for the recall is “Agent &lt;Agent ID&gt; Owned Recall.”</td>
</tr>
<tr>
<td>WaitMsg</td>
<td>If present, an asterisk (*) separates &lt;WaitMsg&gt; from &lt;OpMesg&gt;. &lt;WaitMsg&gt; is a text string defined in the Avaya Proactive Contact job file. It indicates how long the customer has been on hold. &lt;WaitMsg&gt; may 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).</td>
</tr>
<tr>
<td>CallType</td>
<td>Identifies the call direction. If Avaya 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.</td>
</tr>
<tr>
<td>NotifyFieldName</td>
<td>The calling list field name requested by AGTSetNotifyKeyField.</td>
</tr>
<tr>
<td>NotifyFieldData</td>
<td>The notification key field value in the customer’s record.</td>
</tr>
<tr>
<td>FieldName</td>
<td>A calling list field name requested by AGTSetDataField.</td>
</tr>
<tr>
<td>FieldData</td>
<td>The customer record field data.</td>
</tr>
</tbody>
</table>
Commands and notification events

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28885</td>
<td>The agent application is not attached to a job. The call notification event is inappropriate.</td>
</tr>
<tr>
<td>E28900</td>
<td>§ &lt;Message&gt;‡ An Avaya Proactive Contact system internal error occurred. Error message text follows.</td>
</tr>
<tr>
<td>E28906</td>
<td>The agent is not ready for next customer record. Call notification event is inappropriate.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>M00001</td>
<td>§ &lt;OpMsg&gt;[*&lt;WaitMsg&gt;] § &lt;CallType&gt; § &lt;NotifyFieldName&gt;,&lt;NotifyFieldData&gt;‡ Initial notification data message.</td>
</tr>
<tr>
<td>M00001</td>
<td>§ &lt;FieldName&gt;,&lt;FieldData&gt; § &lt;FieldName&gt;, &lt;FieldData&gt;…‡ Additional fields data message.</td>
</tr>
</tbody>
</table>

Examples

Agent Binary to Agent Application

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

0 §
M00001 §
TRANSFER CALL §
INBOUND §
ACCTNUM, 9724973513683608‡
AGTCallNotify N Agent server 6111 0 2§
0 §
M00000‡
AGTCallNotify N Agent server 6111 0 2§
0 §
M00000‡
AgtOwnedRc: 04 J: outbnd1 U: Allid T: 11.26.00 (Agent binary sends this message to
notify Agent 04 of the pending owned recall.) OUTBOUND
ACCTNUM, 55555223331234567
AGTCallNotify N Agent server 6111 0 2§
0 §
M00001‡
Agent 04 Owned Recall (Agent binary sends this message when Agent 04 receives the owned
recall.)
OUTBOUND
ACCTNUM, 55555223331234568

See also

- AGTFinishedItem on page 119
- AGTReadyNextItem on page 205
- AGTSetDataField on page 221
- AGTSetNotifyKeyField on page 224
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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameter

<table>
<thead>
<tr>
<th>ListType</th>
<th>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.</th>
</tr>
</thead>
</table>
Return values

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

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 x03

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

See also

- [AGTAttachJob](#) on page 88
- [AGTDetachJob](#) on page 102
- [AGTSetDataField](#) on page 221
AGTConnHeadset

AGTConnHeadset is a connection command.

Alternate Name

ConnHeadset

Function

Connects a headset to Avaya Proactive Contact using direct-connect, dial-in, or dial-back headset connections. Once this command executes, Avaya 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 x03

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

Data parameters

None.

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28850</td>
<td>Internal Avaya Proactive Contact system error: cannot open a channel to the operator monitor process; unable to connect headset ID. Indicates an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>E28872</td>
<td>Headset is already connected.</td>
</tr>
<tr>
<td>E28873</td>
<td>Headset ID is not reserved. Execute AGTReserveHeadset and retry.</td>
</tr>
<tr>
<td>E28874</td>
<td>A connect headset request is already pending.</td>
</tr>
</tbody>
</table>
E28875 | The headset connect request did not register. Indicates an Avaya Proactive Contact system problem.
---|---
E28876 | The headset is not connected. This message may indicate that there is a problem with the telephone connection to the headset, or there is an Avaya 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 x03
‡ = 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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- [AGTDisconnHeadset](#) on page 108
- [AGTReserveHeadset](#) on page 212
AGTDetachJob

AGTDetachJob is a system command.

Alternate Name

DetachJob

Function

Detaches the agent application from an active job on Avaya 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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28913</td>
<td>No job is attached. There is nothing to detach.</td>
</tr>
<tr>
<td>E28914</td>
<td>Agent is still available for work on the job and calls are still being routed to the agent headset. Execute AGTNoFurtherWork and retry.</td>
</tr>
<tr>
<td>E28915</td>
<td>Not logged off the job yet. The AGTNoFurtherWork command is in the process of executing. Wait for a completion message from Avaya Proactive Contact and retry.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>
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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty).

See also

- AGTAttachJob on page 88
- AGTNoFurtherWork on page 197
AGTDialDigit

AGTDialDigit is a working command.

Alternate name

DialDigit

Function

Causes Avaya 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> <ProclID> <InvokeID> 1§
<Digit> ‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty).

Data parameter

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28866</td>
<td>A telephone line is not available. An open telephone line is only available after the agent receives a call.</td>
</tr>
<tr>
<td>E28867</td>
<td>The telephone line is not off-hook. Place the telephone line in off-hook state and retry.</td>
</tr>
<tr>
<td>E29950</td>
<td>This feature is not available on an Avaya Proactive Contact with CTI system.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S28814</td>
<td>The transfer for the last call is in progress. Retry after the transfer is complete.</td>
</tr>
</tbody>
</table>

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

See also

- AGTHangupCall on page 127
- AGTManualCall on page 187
- AGTRelaseLine on page 209
AGTDialerMode

AGTDialerMode is a working command.

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

Data Parameters

None

Return Values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOFTDIALER</td>
<td>CTI dialer</td>
</tr>
<tr>
<td>DIGITAL</td>
<td>Non-CTI dialer</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

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 x03
‡ = 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 Avaya Proactive Contact reserved headset list. See AGTFreeHeadset on page 121.

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> <ProcID> <InvokeID> 0‡

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

† = Message continues(ETB), ASCII x03

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

Data parameters

None.
Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28873</td>
<td>The headset is not reserved. There is no headset to disconnect.</td>
</tr>
<tr>
<td>E28876</td>
<td>The headset is not connected. There is no headset connection to close.</td>
</tr>
<tr>
<td>E28877</td>
<td>There is a disconnect headset request already pending.</td>
</tr>
<tr>
<td>E28900</td>
<td>§&lt;Message&gt;‡ An Avaya Proactive Contact system internal error occurred. Error message text follows.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S28833</td>
<td>Pending.</td>
</tr>
</tbody>
</table>

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

See also

- AGTAvailWork on page 91
- AGTConnHeadset on page 100
- AGTNoFurtherWork on page 197
- AGTReserveHeadset on page 212
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 Avaya 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 Avaya 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> <ProclID> <InvokeID> 1§
<FileName>‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameter

<table>
<thead>
<tr>
<th>FileName</th>
<th>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 &lt;FileName&gt; parameter is case sensitive.</th>
</tr>
</thead>
</table>

Return value

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>
Commands and notification events

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.

```plaintext
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_msg_cnt = '0'
oper_rec_pos = '-1'
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_updatet ime = 0
inb_worktime = 0
job_callsworked = 0
job_idlecount = 0
job_idletime = 0
job_talktime = 0
job_updatet ime = 0
job_worktime = 0
out_callsworked = 0
out_idlecount = 0
out_idletime = 0
out_talktime = 0
out_updatet ime = 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
iicb_login = 0
Operator AGTSTATE :
attached_job = 1
available_next_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 = listgjg
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
cotime = 22
idle_clock = 827525889
log_time = 0
preview_time = 30
talking_time = 0
port_base = 1750
pqtype = S
jobname = mangjg
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
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
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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

AGTLogIoStart on page 168
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 may 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> <ProcID> <InvokeID> 0‡
‡ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None.

Return value

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

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

See also

AGTEchoOn on page 118
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 may wish to 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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Return value

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

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 on page 116
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 Avaya Proactive Contact after releasing the record.

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

<table>
<thead>
<tr>
<th>CompCode</th>
<th>The two-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.</th>
</tr>
</thead>
</table>

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28866</td>
<td>There is no line available. This error generally indicates the agent is trying to cancel a Managed Dialing call too late.</td>
</tr>
</tbody>
</table>
### Commands and notification events

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28885</td>
<td>The agent application is not attached to a job. Execute AGTAttachJob and retry.</td>
</tr>
<tr>
<td>E28919</td>
<td>The agent is not currently working with a customer record. There is no customer record to release.</td>
</tr>
<tr>
<td>E28947</td>
<td>Completion code is invalid. Retry command with a code returned by AGTListKeys for the current job.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S28833</td>
<td>Pending.</td>
</tr>
</tbody>
</table>

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

### See also

- [AGTCallNotify](#) on page 94
- [AGTListKeys](#) on page 161
- [AGTReadyNextItem](#) on page 205
- [AGTReleaseLine](#) on page 209
AGTFreeHeadset

AGTFreeHeadset is a connection command.

Alternate name

FreeHeadset

Function

Frees the reserved headset ID by:

● Removing the ID from the appropriate tables on Avaya 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> <ProcID> <InvokeID> 0‡

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

Data parameters

None.

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28873</td>
<td>There is no headset reserved. There is nothing to free.</td>
</tr>
<tr>
<td>E28879</td>
<td>The headset is not disconnected. Disconnect the headset and retry.</td>
</tr>
</tbody>
</table>
Commands and notification events

<table>
<thead>
<tr>
<th>M00000</th>
<th>Complete.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S28833</td>
<td>Pending.</td>
</tr>
</tbody>
</table>

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§

See also

- AGTDisconnHeadset on page 108
- AGTReserveHeadset on page 212
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 Avaya Proactive Contact systems using OLIC cards.

Format
AGTGetHeadsetVol C <Client> <ProclID> <InvokeID> 0 ‡

Data parameters
None.

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28850</td>
<td>Cannot open a channel to the operator monitor process. Indicates an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>E28851</td>
<td>There is no response from the operator monitor process. Indicates an Avaya system problem.</td>
</tr>
<tr>
<td>E28866</td>
<td>A telephone line not available. Retry with an open telephone line.</td>
</tr>
<tr>
<td>E28867</td>
<td>The open telephone line is not off-hook. Retry with an off-hook telephone line.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>
Commands and notification events

<table>
<thead>
<tr>
<th>M00001</th>
<th>§ &lt;Ear&gt; § &lt;Mouth&gt;‡</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data message. The current volume settings for the headset earphone and microphone. Range for settings is from one to eight.</td>
</tr>
</tbody>
</table>

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

See also

- [AGTAdjustHeadset](#) on page 84
- [AGTHangupCall](#) on page 127
- [AGTManualCall](#) on page 187
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 may move to the field. 0 = cursor may not move to the field.
3. Locked. 1 = locks the field. 0 = lets agents enter data.
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 may be moved to the field and the agent may 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

```
CLIENT --> 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"^^F, 9,20, 0,"ZIPCODE:0:1:C::0:0"^^F,47,20, 0,"INTERNAL:0:1:C::0:0"^^F,70,20,
   0,"EXTERNAL:0:1:C::0:0"^^F,1,11, 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,
   , 9,"External:"^^L,36,22, 8,"Behavior:"^^L,45,22, 6,"Score:"^^L
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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None.

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28866</td>
<td>A telephone line is not available. There is no call to hang up.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

Examples

Agent Application to Agent Binary

AGTHangupCall C COriginator_ID 11111 75 0‡

Agent Binary to Agent Application
Commands and notification events

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

See also

- AGTDialDigit on page 104
- AGTManualCall on page 187
- AGTReleaseLine on page 209
AGTHeadsetConnBroken

AGTHeadsetConnBroken is a notification event message from the server.

Function

Notifies the agent application that no agent headset connection exists. The headset may have become physically disconnected or the agent may 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.

Note:

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

Data parameters

None.

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28880</td>
<td>Headset connection is broken.</td>
</tr>
<tr>
<td>E28881</td>
<td>Headset is reconnected.</td>
</tr>
</tbody>
</table>
| E28900 | § <message>‡
An Avaya Proactive Contact system internal error occurred. Error message text follows. |

Examples

AGTHeadsetConnBroken N Agent server 22486 0 2§
See also

- [AGTConnHeadset](#) on page 100
- [AGTDisconnHeadset](#) on page 108
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 x03

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

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28866</td>
<td>Telephone line is not available. There is no open telephone line.</td>
</tr>
<tr>
<td>E28867</td>
<td>The open telephone line is not off-hook. Retry with an off-hook telephone</td>
</tr>
<tr>
<td></td>
<td>line. This message may mean the agent has already placed the call on hold.</td>
</tr>
<tr>
<td>E29950</td>
<td>This feature is not available on an Avaya Proactive Contact with CTI system.</td>
</tr>
</tbody>
</table>
Commands and notification events

<table>
<thead>
<tr>
<th>M00000</th>
<th>Complete.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S28814</td>
<td>Transfer is in progress. Avaya Proactive Contact cannot place the call on hold.</td>
</tr>
</tbody>
</table>

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

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

† = Message continues(ETB), ASCII x03

See also

- [AGTFinishedItem](#) on page 119
- [AGTReleaseLine](#) on page 209
- [AGTU unholdCall](#) on page 240
AGTHookflashLine

AGTHookflashLine is a working command.

Alternate Name

HookflashLine

Function

Sends a hookflash and places a manual call.
Use it to transfer calls on Avaya 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 may 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> <ProclID> <InvokeID> 1§<PhoneNumber> ‡
AGTHookflashLine C <Client> <ProclID> <InvokeID> 1§ ‡
Commands and notification events

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 Avaya Proactive Contact system. |

Return Values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28866</td>
<td>Telephone line is not available. Avaya Proactive Contact cannot place the call. Retry later.</td>
</tr>
<tr>
<td>E28867</td>
<td>Telephone line not off-hook. Retry with an off-hook open line.</td>
</tr>
<tr>
<td>E29950</td>
<td>This feature is not available on an Avaya Proactive Contact with CTI system.</td>
</tr>
<tr>
<td>S28814</td>
<td>Transfer is in progress. Retry after the transfer is complete.</td>
</tr>
</tbody>
</table>

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

See Also

- AGTManualCall on page 187
- AGTTransferCall on page 238
AGTlicbAbort

AGTlicbAbort is a notification event message from the server.

**Alternate name**

IicbAbort

**Function**

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

Avaya 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 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)
Commands and notification events

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

See also

- AGTlicbFeNotif on page 137
- AGTlicbOffline on page 139
- AGTlicbOnline on page 141
AGTlicbFeNotif

AGTlicbFeNotif is a notification event message from the server.

Alternate name

licbFeNotif

Function

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

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

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Commands and notification events

<table>
<thead>
<tr>
<th>S28971</th>
<th>Standby to take outbound calls... Acquisition to outbound is pending.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S28972</td>
<td>Standby to take inbound calls... Release to inbound is pending.</td>
</tr>
</tbody>
</table>

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

See also

- [AGTlicbOffline](#) on page 139
- [AGTlicbOnline](#) on page 141
AGTlicbOffline

AGTlicbOffline is a notification event message from the server.

Alternate name

licbOffline

Function

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

Avaya 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.

Format

AGTlicbOffline N <Agent server> <ProcID> <InvokeID> 2§
0 § M00000‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

Examples

Agent Binary to Agent Application

AAGTlicbOffline N Agent server 9736 0 2§
0 §
Commands and notification events

\[ M00000 \]

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- AGTlicbFeNotif on page 137
- AGTlicbOnline on page 141
AGTLicbOnline

AGTLicbOnline is a notification event message from the server.

Alternate name

IicbOnline

Function

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

Avaya Proactive Contact sends this message to inform the agent application that Avaya 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> <ProcID> <InvokeID> 2§
  0 § M00000‡
  § = Message data separator, ASCII x1E (socket) or comma (tty)
  † = Message continues(ETB), ASCII x03
  ‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

Examples

Agent Binary to Agent Application

AAGTLicbOnline N Agent server 2478 0 2§
  0 §
Commands and notification events

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- AGTlibcFeNotif on page 137
- AGTlibcOffline on page 139
- AGTLogonAc on page 182
AGTJobEnd

AGTJobEnd is a notification event message from the server.

Alternate name

JobEnd

Function

The Avaya 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> <ProclID> <InvokeID> 2§
0 § M00000‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None

Return values

None

Examples

Agent Binary to Agent Application

AAGTJobEnd N Agent server 17970 0 2§
0 §
M00000‡
Commands and notification events

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- AGTAttachJob on page 88
- AGTDetachJob on page 102
- AGTJobTransLink on page 145
AGTJobTransLink

AGTJobTransLink is a notification event message from the server.

Alternate name

JobTransLink

Function

This message only appears on Avaya 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 x03
‡ = 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 may not include special characters or embedded spaces. |
Commands and notification events

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28900</td>
<td>§ &lt;message&gt;‡ An Avaya Proactive Contact system internal error occurred. Error message text follows.</td>
</tr>
<tr>
<td>E28917</td>
<td>No job is attached. There is no current job that could have a job linked to it.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

Examples

Agent Binary to Agent Application

```
AAGTJobTransLink N Agent server 17970 0 3§
  0 §
  M00000 §
  outbnd1‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- [AGTAttachJob](#) on page 88
- [AGTDetachJob](#) on page 102
- [AGTJobEnd](#) on page 143
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 <ProcID> <InvokeID> 3§
0 § M00000 § <JobName>‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

<table>
<thead>
<tr>
<th>&lt;JobName&gt;</th>
<th>The value is the name of the job the agent should transfer to. Use up to 19 characters, 7-bit USASCII. The &lt;JobName&gt; is case sensitive and may not include special characters or embedded spaces.</th>
</tr>
</thead>
</table>

Commands and notification events

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28900</td>
<td>§ &lt;Message&gt; ‡ An Avaya Proactive Contact system internal error occurred. Error message text follows.</td>
</tr>
<tr>
<td>E28901</td>
<td>Agent is not available for work. The AGTNoFurtherWork command initiated by agent is not necessary.</td>
</tr>
<tr>
<td>E28917</td>
<td>No job is attached. The transfer request is inappropriate.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

Examples

Agent Binary to Agent Application

AGTJobTransRequest N Agent server 4880 0 3§  
0 $  
M00001 $  
outgjg‡  
AGTJobTransRequest N Agent server 4880 0 2§  
0 $  
M00000 $  

§ = Message data separator, ASCII x1E (socket) or comma (tty)  
† = Message continues(ETB), ASCII x03  
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- [AGTAttachJob](on page 88)
- [AGTDetachJob](on page 102)
- [AGTNoFurtherWork](on page 197)
AGTListCallbackFmt

AGTListCallbackFmt is a working command.

Alternate name

ListCallbackFmt

Function

Use this command to determine the date format used when sending date information to Avaya 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 Avaya 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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28868</td>
<td>No recalls during inbound jobs. The current attached job is an inbound job. There is no recall format.</td>
</tr>
<tr>
<td>E28885</td>
<td>The agent application is not attached to a job. Execute AGTAttachJob and retry.</td>
</tr>
</tbody>
</table>
Commands and notification events

<table>
<thead>
<tr>
<th>M00000</th>
<th>Complete.</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00001</td>
<td>§ &lt;Format&gt; § &lt;Phones&gt;‡Data message. &lt;Format&gt; is the date format used on Avaya Proactive Contact. * &lt;Phones&gt; is the number of phones in the customer record available for recall. This provides a range (from 1 to &lt;Phones&gt;) of index values to use with AGTSetCallback.</td>
</tr>
</tbody>
</table>

Examples

**Agent Application to Agent Binary**

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

See also

[AGTSetCallback](on page 216)
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 may execute any time after AGTLogon executes.

Format

AGTListCallFields C <Client> <ProcID> <InvokeID> 1§
/ListName‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)
‡ = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

<table>
<thead>
<tr>
<th>ListName</th>
<th>Enter the name of the calling list. Use alphanumeric values, 7-bit USASCII. Calling list names are case sensitive.</th>
</tr>
</thead>
</table>
Commands and notification events

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E00518</td>
<td>&lt;agent&gt; &lt;ListName&gt; There is no calling list with &lt;ListName&gt; on the system. The agent entry is the name of the process on Avaya Proactive Contact returning the error (it will always be agent). &lt;ListName&gt; is the name submitted by the client process.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>M00001</td>
<td>$ &lt;FieldName&gt;,&lt;FieldLength&gt;,&lt;FieldType&gt;,F $ &lt;FieldName&gt;,&lt;FieldLength&gt;,&lt;FieldType&gt;,F... Data message. The &lt;FieldType&gt; is C (character), N (numeric), $(currency), D (date), or T (time). The F value is a placeholder for future use.</td>
</tr>
</tbody>
</table>

Examples

Agent Application to Agent Binary

AAGTListCallFields C COiginator_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 § SWFLAG,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
  $ CRELINE,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 §
  DSCNT,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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- [AGTListDataFields](#) on page 156
- [AGTReadField](#) on page 202
- [AGTSetDataField](#) on page 221
- [AGTSetNotifyKeyField](#) on page 224
- [AGTUpdateField](#) on page 242
**AGTListCallLists**

AGTListCallLists is a system command.

**Alternate name**

ListCallLists

**Function**

Lists all calling lists on Avaya 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 x03

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

**Data parameters**

None

**Return values**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>M00001</td>
<td>§ &lt;ListName1&gt; § &lt;ListName2&gt;...‡</td>
</tr>
</tbody>
</table>

**Examples**

**Agent Application to Agent Binary**

AAGTListCallLists C COriginator_ID 11111 2 0‡
Agent Binary to Agent Application

AGTListCallLists D Agent server 17939 2 7§
0 §
M00001 §
inbnd1 §
inbrpt §
latel1 §
list1 §
list2‡
AGTListCallLists R Agent server 17939 2 2§
0 §
M00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

AGTListCallLists on page 154
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> <ProcID> <InvokeID> 1§
<ListType> ‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

<table>
<thead>
<tr>
<th>ListType</th>
<th>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.</th>
</tr>
</thead>
</table>

156  Avaya Proactive Contact Agent API Reference
Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28885</td>
<td>Agent application is not attached to a job. Execute AGTAttachJob and then retry.</td>
</tr>
<tr>
<td>E28891</td>
<td>Must specify I (inbound) or O (outbound) calling list type. Retry command with &lt;ListType&gt; parameter included.</td>
</tr>
<tr>
<td>E28892</td>
<td>No inbound calling list fields are available. Retry using &lt;ListType&gt; O.</td>
</tr>
<tr>
<td>E28893</td>
<td>No outbound calling list fields are available. Retry using &lt;ListType&gt; I.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>M00001</td>
<td>Data message. The &lt;FieldType&gt; is C (character), N (numeric), $(currency), D (date), or T (time). The F value is a placeholder for future use.</td>
</tr>
</tbody>
</table>

Examples

Agent Application to Agent Binary

```plaintext
AGTListDataFields C COriginator_ID 11111 19 1$
  0$
```

Agent Binary to Agent Application

```plaintext
AGTListDataFields D Agent server 15395 19 42$
  0$
M00001$
  *
  0$
M00000$
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)
Commands and notification events

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

See also

- AGTAttachJob on page 88
- AGTReadField on page 202
- AGTSetDataField on page 221
- AGTSetNotifyKeyField on page 224
- AGTUpdateField on page 242
AGTListJobs

AGTListJobs is a system command.

Alternate name

ListJobs

Function

Returns the job type, name, and status for each job defined on Avaya 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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

<table>
<thead>
<tr>
<th>JobType</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>(I) inactive or (A) active</td>
</tr>
</tbody>
</table>

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28884</td>
<td>The agent binary was unable to access Avaya Proactive Contact shared memory. This usually indicates an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>E70003</td>
<td>Unknown job type. Retry with a job type shown above.</td>
</tr>
</tbody>
</table>
Commands and notification events

<table>
<thead>
<tr>
<th>M00000</th>
<th>Complete.</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00001</td>
<td>§ &lt;JobType&gt;, &lt;JobName&gt;, &lt;Status&gt; § &lt;JobType&gt;, &lt;JobName&gt;, &lt;Status&gt;…‡Data message containing requested job information. Job status appears as a single digit, I for inactive or A for active.</td>
</tr>
</tbody>
</table>

Examples

Agent Application to Agent Binary

```
AGTListJobs C COiginator_ID 11111 18 1§
A‡
```

Agent Binary to Agent Application

```
AGTListJobs D Agent server 15395 18 8 §
0 §
M00001 §
B,blend1,I §
I,inbndl,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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- [AGTAttachJob](#) on page 88
- [AGTLogon](#) on page 180
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 may return incorrect values to Avaya Proactive Contact.

Availability

Execute this command any time after executing AGTAttachJob.

Format

AGTListKeys C <Client> <ProcID> <InvokeID> 0 ‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E00518</td>
<td>Unable to open the keys file. Indicates an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>E28885</td>
<td>Not attached to a job. Execute AGTAttachJob and retry the command.</td>
</tr>
</tbody>
</table>
Commands and notification events

<table>
<thead>
<tr>
<th>M00000</th>
<th>Complete.</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00001</td>
<td>Data message. Completion codes are two digits. Call centers define call completion codes during specification of their Avaya Proactive Contact system. The descriptions are from the Avaya Proactive Contact Compcode.cfg file. Telephone script labels are labels found in Avaya Proactive Contact Telephony.spt file. The system associates the labels with the respective call completion codes.</td>
</tr>
</tbody>
</table>

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 x03

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

See also

- [AGTAttachJob](#) on page 88
- [AGTFinishedItem](#) on page 119
- [AGTReleaseLine](#) on page 209
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 Avaya 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 32.

Availability

This command is available any time after AGTLogon executes.

Format

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

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

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S70000</td>
<td>&lt;JobName&gt;Agent is on a call. Working on &lt;JobName&gt;, currently working with a customer record.</td>
</tr>
<tr>
<td>S70001</td>
<td>&lt;JobName&gt;Agent is ready for a call. Working on &lt;JobName&gt;, currently waiting for next customer record.</td>
</tr>
</tbody>
</table>
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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- **AGTAttachJob** on page 88
- **AGTAvailWork** on page 91
- **AGTFinishedItem** on page 119
- **AGTNoFurtherWork** on page 197
- **AGTReadyNextItem** on page 205
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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28885</td>
<td>Not attached to a job. Attach an active Unit Work List job and retry.</td>
</tr>
<tr>
<td>E28886</td>
<td>The attached job is not a Unit Work List job. Change attached job and retry.</td>
</tr>
</tbody>
</table>
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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- AGTAttachJob on page 88
- AGTSetUnit on page 228
AGTLogIoStart

AGTLogIoStart is a system command.

Alternate name

LogIoStart

Function

Creates a file on the Avaya Proactive Contact system named <AgentName>_API.trans. It contains all the messages between the agent application and Avaya 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 Avaya Proactive Contact Agent API on page 23.

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.

Note:

In the Service Pack 1 release, binaries have been modified to remove /tmp from in front of the filename specified for DBGOUTPUT. This applies the trans.log files created when you send the AGTLogIoStart command.

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

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**

AGTLogIoStart R Agent server 5846 26 2§

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

† = Message continues(ETB), ASCII x03

‡ = 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
,blendgjg
SERVER<-- AGTAttachJob R Agent server 8617 5 2
^^0^^M00000
Commands and notification events

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
^^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
AGTLogIoStart

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
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
 ^^0"^S28833
SERVER<<-- AGTReserveHeadset P Agent server 8618 2 2
 ^^0"^M00000
SERVER<<-- AGTReserveHeadset R Agent server 8618 2 2
 ^^0"^M00000
CLIENT--> AGTConnHeadset C COriginator_ID 111111 3 0
 ^^0"^S28833
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
 ^^0"^M00000
CLIENT--> AGTAttachJob C COriginator_ID 111111 5 1
 ,blendgig
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
 ^^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
See also

- [AGTDumpData](#) on page 111
- [AGTLogloStop](#) on page 174
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> <ProcID> <InvokeID> 0‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = 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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

AGTLogIoStart on page 168
AGTLogoff

AGTLogoff is a connection command.

Alternate name

Logoff

Function

Exits Avaya 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 terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28916</td>
<td>There is a job attached. Detach the job and retry.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S28833</td>
<td>Pending.</td>
</tr>
</tbody>
</table>

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

See also

- AGTDetachJob on page 102
- AGTLogon on page 180
AGTLogoffAcd

AGTLogoffAcd is a connection command.

Alternate name

LogoffAcd

Function

Removes the agent registration with Avaya Proactive Contact Dispatcher process. Removing the registration logs the agent out of the ACD and disconnects the agent's headset. After AGTLogoffAcd executes, Avaya Proactive Contact cannot acquire the agent.

Use this command to remove the agent from Agent Blending before logging the agent out of Avaya Proactive Contact.

After using this command to log out of Agent Blending and the ACD, the agent may 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

AGTLogoffAcd C <Client> <ProcId> <InvokeID> 0‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None
Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S28833</td>
<td>Pending</td>
</tr>
</tbody>
</table>

Examples

Agent Application to Agent Binary

```
AGTLogoffAcd C COriginator_ID 11111 122 0†
```

Agent Binary to Agent Application

```
AGTLogoffAcd P Agent server 17939 122 2§
   0 §
   S28833‡
AGTLogoffAcd R Agent server 17939 122 2§
   0 §
   M00000‡
```

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- AGTConnHeadset on page 100
- AGTDetachJob on page 102
- AGTFreeHeadset on page 121
- AGTLogoff on page 176
- AGTLogoffAcd on page 178
- AGTReserveHeadset on page 212
AGTLogon

AGTLogon is a connection command.

Alternate name

Logon

Function

Verifies a user name and password on Avaya Proactive Contact. AGTLogon also registers with the Agent_count binary that controls licenses on the system.

Availability

This command is the only command available to the agent application when it first connects to Avaya Proactive Contact. Execute this command before issuing any other commands.

Format

AGTLogon C <Client> <ProcID> <InvokeID> 2
§ <AgentName> § <Password> ‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

<table>
<thead>
<tr>
<th>AgentName</th>
<th>Enter the user name of the calling agent. Use up to 19 characters, 7-bit USASCII character set. Do not include embedded spaces.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>Enter the agent’s password. Use two to eight characters, 7-bit USASCII character set. Do not include embedded spaces.</td>
</tr>
</tbody>
</table>

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28812</td>
<td>&lt;AgentName&gt; is logged ion to another application. The agent may be logged on at another location. Retry logon with a different &lt;AgentName&gt;.</td>
</tr>
</tbody>
</table>
### AGTLogon

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28925</td>
<td>This agent application is already logged in to the system from this session. To change the <code>&lt;AgentName&gt;</code>, execute AGTLogoff and retry.</td>
</tr>
<tr>
<td>E28926</td>
<td>Invalid login. Indicates that the user name or password is invalid.</td>
</tr>
<tr>
<td>E50100</td>
<td>The maximum number of agents allowed by the Agent_count binary are already logged on to the system.</td>
</tr>
<tr>
<td>E70012</td>
<td>The agent current password is no longer valid. Enter a new password to continue the log in.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S28833</td>
<td>Pending.</td>
</tr>
</tbody>
</table>

### Examples

**Agent Application to Agent Binary**

```plaintext
AGTLogon C COriginator_ID 111111 1 2§
   01 §
   01†
```

**Agent Binary to Agent Application**

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

### See also

- [AGTLogoff](#) on page 176
AGTLogonAcd

AGTLogonAcd is a connection command.

Alternate name

LogonAcd

Function

Registers the agent with the Dispatcher process on Avaya 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> <ProclId> <InvokeID> 2§
<Extension> § <PBX ID>‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

<table>
<thead>
<tr>
<th>Extension</th>
<th>The ACD extension to associate with the agent’s telephone. Use a numeric value, up to 20 characters.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBX ID</td>
<td>The identification of the PBX (switch) used for Agent Blending. Default value is 1. Value given here must match the DBKGRUP parameter in the Avaya Proactive Contact master.cfg file.</td>
</tr>
</tbody>
</table>
Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28872</td>
<td>The agent is already logged on to ACD.</td>
</tr>
<tr>
<td>E28873</td>
<td>Headset is not reserved. Reserve a headset identification with AGTReserveHeadset before executing this command.</td>
</tr>
<tr>
<td>E28950</td>
<td>The specified extension is not one of the ACD extensions defined on Avaya Proactive Contact.</td>
</tr>
<tr>
<td>E28951</td>
<td>The specified extension is in use by another agent.</td>
</tr>
<tr>
<td>E28952</td>
<td>Cannot add another agent. The maximum number of ACD agents are already logged on.</td>
</tr>
<tr>
<td>E28953</td>
<td>Could not read the file containing the ACD extensions. Indicates an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>E28954</td>
<td>Duplicate logon - Please try again. Indicates that the agent user name is already logged in.</td>
</tr>
<tr>
<td>E28955</td>
<td>The Agent Blending Dispatcher process is not running.</td>
</tr>
<tr>
<td>E28956</td>
<td>ACD logon error. Dispatcher returned an “unknown” response to the agent application.</td>
</tr>
<tr>
<td>E70011</td>
<td>Agent Blending is not available on this Avaya Proactive Contact system. This system cannot work with ACD agents.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S28833</td>
<td>Pending.</td>
</tr>
</tbody>
</table>

Examples

**Agent Application to Agent Binary**

```
AGTLogonACD C COriginator_ID 111111 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 x03
Commands and notification events

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

See also

AGTLogoffAcd on page 178
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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28885</td>
<td>Not attached to a job. Attach a job and retry command.</td>
</tr>
<tr>
<td>E28900</td>
<td>§ &lt;message&gt;‡ An Avaya Proactive Contact system internal error occurred. Error message text follows.</td>
</tr>
<tr>
<td>E28901</td>
<td>Agent is not available for work. Retry after executing AGTAvailWork.</td>
</tr>
<tr>
<td>E28907</td>
<td>Attached job is not a Managed Dialing job. Command is only available for Managed Dialing jobs.</td>
</tr>
<tr>
<td>E28908</td>
<td>Agent is not previewing a customer record. Retry after receiving a record to preview.</td>
</tr>
</tbody>
</table>
### Commands and notification events

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28909</td>
<td>Managed call is already complete. The AGTManagedCall command is already being executed. May occur when agent executes the command just as the time-out elapses.</td>
</tr>
<tr>
<td>E28910</td>
<td>Managed call already canceled or complete. Either an AGTFinishedItem, AGTReleaseLine or AGTManagedCall command is already in the process of executing. May 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.</td>
</tr>
<tr>
<td>E28911</td>
<td>Managed call already canceled. AGTFinishedItem is already executing.</td>
</tr>
<tr>
<td>E28964</td>
<td>Agent phone is busy. Release phone line before proceeding.</td>
</tr>
<tr>
<td>E28965</td>
<td>Softdialer link is down. Please wait until the link is up.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>M00001</td>
<td>$&lt;\text{CallStatus}&gt;$† Data message. $&lt;\text{CallStatus}&gt;$ should always be (CONNECT).</td>
</tr>
<tr>
<td>S28833</td>
<td>Pending. Call is being placed.</td>
</tr>
</tbody>
</table>

### Examples

**Agent Application to Agent Binary**

```plaintext
AGTManagedCall C COriginator_ID 11111 1 0†
```

**Agent Binary to Agent Application**

```plaintext
AGTManagedCall P Agent server 2570 1 2§
  $ S28833‡
AGTManagedCall D Agent server 2570 1 3§
  $ M00001 § (CONNECT)‡
AGTManagedCall R Agent server 2570 1 2§
  $ M00000‡
```

$ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

### See also

- [AGTFinishedItem](#) on page 119
- [AGTPreviewRecord](#) on page 199
- [AGTReadyNextItem](#) on page 205
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 on page 238 and AGTHookflashLine on page 133.

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 x03
‡ = 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 Avaya Proactive Contact system. |
Commands and notification events

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
</table>
| E28843  | § <PhoneNumber>‡  
The <PhoneNumber> is invalid. Either the format of the number does not match the standard phone format on Avaya Proactive Contact or the telephone number itself is not valid. |
| E28866  | Telephone line is not available. Avaya Proactive Contact cannot place an outbound call. Retry after receiving a call from a customer.          |
| 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 Avaya 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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- AGTDialDigit on page 104
- AGTHangupCall on page 127
- AGTHookflashLine on page 133
- AGTTransferCall on page 238
**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. Avaya Proactive Contact places the customer call in the queue for the new job or connects the customer to an available agent.

If Avaya Proactive Contact uses ANI/DNIS information, the transferred call carries the outbound number Avaya 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, Avaya 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.

<table>
<thead>
<tr>
<th>FROM</th>
<th>Agent Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job Type</td>
<td>Agent Type</td>
</tr>
<tr>
<td>Outbound</td>
<td>Outbound</td>
</tr>
<tr>
<td>Outbound</td>
<td>Outbound</td>
</tr>
</tbody>
</table>
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 Avaya 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 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 Avaya 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 Avaya Proactive Contact *.job file for the original job contains a default transfer job name. |

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28866</td>
<td>No phone line is available. Avaya Proactive Contact cannot place the transfer call.</td>
</tr>
<tr>
<td>E28867</td>
<td>Line is not off-hook. There is no telephone call to transfer.</td>
</tr>
<tr>
<td>E28942</td>
<td>Transfer job is not available. The &lt;JobName&gt; specified is not running, does not exist, or is not an inbound or blend job.</td>
</tr>
<tr>
<td>E70007</td>
<td>Cannot transfer an inbound call. AGTMoFlashBlind can only transfer outbound calls.</td>
</tr>
<tr>
<td>E70008</td>
<td>Must specify a transfer job. There is no default transfer job configured on Avaya Proactive Contact, so you must include a transfer &lt;JobName&gt; with the command.</td>
</tr>
<tr>
<td>E70009</td>
<td>,&lt;process name&gt; Unable to send a message to &lt;process name&gt;. Internal Avaya Proactive Contact error; agent cannot send messages.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S28814</td>
<td>Trunk-to-trunk transfer is already in progress. May mean the agent previously executed AGTTransferCall.</td>
</tr>
<tr>
<td>S28833</td>
<td>Pending</td>
</tr>
</tbody>
</table>

Examples

Agent Application to Agent Binary

AGTMoFlashBlind C COriginator_ID 11111 129 1§outbndl‡

Agent Binary to Agent Application

AGTMoFlashBlind P Agent server 8735 129 1§S28833‡
AGTMoFlashBlind R Agent server 8735 129 2§0 §
Commands and notification events

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

● AGTHookflashLine on page 133
● AGTMoFlashSupv on page 193
● AGTTransferCall on page 238
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 Avaya Proactive Contact uses ANI/DNIS information, the transferred call carries the outbound number Avaya 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 Avaya 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 Avaya 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.

<table>
<thead>
<tr>
<th>FROM</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Job Type</td>
<td>Agent Type</td>
</tr>
<tr>
<td>Outbound</td>
<td>Outbound</td>
</tr>
<tr>
<td>Outbound</td>
<td>Outbound</td>
</tr>
<tr>
<td>Blend</td>
<td>Outbound</td>
</tr>
<tr>
<td>Blend</td>
<td>Blend</td>
</tr>
<tr>
<td>Managed</td>
<td>Managed</td>
</tr>
<tr>
<td>Managed</td>
<td>Managed</td>
</tr>
</tbody>
</table>
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 Avaya Proactive Contact begins executing AGTMoFlashSupv, it places the customer on hold. In the simplest case, Avaya 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 Avaya Proactive Contact connects the agents.

If the transferring agent wishes to cancel the transfer, the transferring agent executes AGTHangupCall. This causes Avaya 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 Avaya 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.

<table>
<thead>
<tr>
<th>TO</th>
<th>Job Type</th>
<th>Agent Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inbound</td>
<td>Inbound</td>
</tr>
<tr>
<td>Blend</td>
<td>Blend or Inbound</td>
<td></td>
</tr>
<tr>
<td>Inbound</td>
<td>Inbound</td>
<td></td>
</tr>
<tr>
<td>Blend</td>
<td>Blend or Inbound</td>
<td></td>
</tr>
<tr>
<td>Inbound</td>
<td>Inbound</td>
<td></td>
</tr>
<tr>
<td>Blend</td>
<td>Blend or Inbound</td>
<td></td>
</tr>
</tbody>
</table>
Format

AGTMoFlashSupv C <Client> <ProcID> <InvokeID> 1§
§ = Message data separator, ASCII x1E (socket) or comma (tty)
§ = Message continues(ETB), 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 Avaya Proactive Contact to receive the transferred 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 Avaya Proactive Contact *.job file for the original job contains a default transfer job name. |

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28866</td>
<td>No phone line is available. Avaya Proactive Contact does not have a telephone line available to place the transfer call.</td>
</tr>
<tr>
<td>E28867</td>
<td>Line is not off-hook. There is no call to transfer.</td>
</tr>
<tr>
<td>E28868</td>
<td>The transfer failed. Try again later. There is no agent available on the specified job to take the transfer call.</td>
</tr>
<tr>
<td>E28942</td>
<td>Transfer job is not available. The &lt;JobName&gt; specified does not exist, is not active, or is an inbound or blend job.</td>
</tr>
<tr>
<td>E70007</td>
<td>Cannot transfer an inbound call. AGTMoFlashSupv can only transfer outbound calls.</td>
</tr>
<tr>
<td>E70008</td>
<td>Must specify a transfer job. There is no default transfer job configured on Avaya Proactive Contact. You must include a transfer &lt;JobName&gt; with the command.</td>
</tr>
<tr>
<td>E70009</td>
<td>&lt;process name&gt; Unable to send a message to &lt;process name&gt;. Internal Avaya Proactive Contact error; agent cannot send messages.</td>
</tr>
</tbody>
</table>
Commands and notification events

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E70010</td>
<td>Conference is in progress. The agent executed the command a third time after the conference call began.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S28814</td>
<td>Trunk-to-trunk transfer is already in progress. May mean the agent previously executed AGTTransferCall.</td>
</tr>
<tr>
<td>S28833</td>
<td>Pending.</td>
</tr>
</tbody>
</table>

Examples

Agent Application to Agent Binary

AGTMoFlashSupv C COriginator_ID 11111 129 1§
  outbnd1‡

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

See also

- AGTHookflashLine on page 133
- AGTMoFlashSupv on page 193
- AGTTransferCall on page 238
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 may 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, Avaya 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 may execute any time the agent is available for work.

If the agent is currently working with a customer record or a call, Avaya Proactive Contact holds the request until AGTFinishedItem executes.

Format

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

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

† = Message continues(ETB), ASCII x03

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

Data parameters

None
Commands and notification events

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28917</td>
<td>No job is attached. There is nothing to log out.</td>
</tr>
<tr>
<td>E28918</td>
<td>Not available for work on the job. Agent is not logged in to a job; nothing to log off.</td>
</tr>
<tr>
<td>E28967</td>
<td>An agent attempted to log out of an agent-owned recall without logging out of the job from which the agent transferred.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S28833</td>
<td>Pending. This state can last through one or more customer calls.</td>
</tr>
</tbody>
</table>

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

See also

- [AGTAttachJob](#) on page 88
- [AGTAvailWork](#) on page 91
- [AGTDetachJob](#) on page 102
- [AGTFinishedItem](#) on page 119
AGTPreviewRecord

AGTPreviewRecord is a notification event message from the server.

Alternate name

PreviewRecord

Function

Notifies the agent application that Avaya 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 may occur instead of the AGTPreviewRecord data messages.

Format

AGTPreviewRecord N <Agentserver> <ProclD> <InvokeID> 6§
0 § M00001 §<OpMesg> § <CallType>
§<NotifyFieldName>, <FieldData>‡

AGTPreviewRecord N <Agentserver> <ProclD> <InvokeID> 3§
0 §M00001 §<FieldName>,<FieldData> §
<FieldName>,<FieldData>...‡
Commands and notification events

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpMesg</td>
<td>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 Avaya Proactive Contact system defines the field. The text string “(Preview)” always follows the information.</td>
</tr>
<tr>
<td>CallType</td>
<td>Always MANAGED. Only Managed Dialing jobs send preview records.</td>
</tr>
<tr>
<td>NotifyFieldName</td>
<td>Field name from the calling list which the agent application requested by executing AGTSetNotifyKeyField.</td>
</tr>
<tr>
<td>NotifyFieldData</td>
<td>Value of notification key field from the customer record.</td>
</tr>
<tr>
<td>FieldName</td>
<td>A field name from the calling list which the agent application requested by executing AGTSetDataField.</td>
</tr>
<tr>
<td>FieldData</td>
<td>Value of data field from the customer record.</td>
</tr>
</tbody>
</table>

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28885</td>
<td>The agent application is not attached to a job. Preview record event is inappropriate.</td>
</tr>
<tr>
<td>E28900</td>
<td>§ &lt;message&gt;‡ An Avaya Proactive Contact system internal error occurred. Error message text follows.</td>
</tr>
<tr>
<td>E28906</td>
<td>The agent is not ready for next customer record. Preview record event is inappropriate.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>M00001</td>
<td>§ &lt;OpMesg&gt; § &lt;CallType&gt; § &lt;NotifyFieldName&gt;, &lt;FieldData&gt;‡ Initial preview data message.</td>
</tr>
<tr>
<td>M00001</td>
<td>§ &lt;FieldName&gt;,&lt;FieldData&gt; § &lt;FieldName&gt;,&lt;FieldData&gt;...‡ Additional preview field data message.</td>
</tr>
</tbody>
</table>

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

See also

- [AGTManagedCall](#) on page 185
- [AGTReadyNextItem](#) on page 205
- [AGTSetDataField](#) on page 221
- [AGTSetNotifyKeyField](#) on page 224
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 may 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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

<table>
<thead>
<tr>
<th>ListType</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>A field name from the calling list, as shown by AGTListDataFields. Up to 19 characters, alphanumeric, case sensitive, no embedded spaces.</td>
</tr>
</tbody>
</table>
Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28885</td>
<td>Not attached to a job. Retry with an active job and an open customer record.</td>
</tr>
<tr>
<td>E28891</td>
<td>Must specify inbound or outbound operation. Retry with &lt;ListType&gt; set to I or O.</td>
</tr>
<tr>
<td>E28892</td>
<td>No inbound calling list fields available. Retry with &lt;ListType&gt; set to O.</td>
</tr>
<tr>
<td>E28893</td>
<td>No outbound calling list fields available. Retry with &lt;ListType&gt; set to I.</td>
</tr>
<tr>
<td>E28894</td>
<td>&lt;FieldName&gt; not found in calling list. Retry with valid field name for the calling list.</td>
</tr>
<tr>
<td>E28912</td>
<td>Customer record is not available. The agent is not working with a customer record.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>M00001</td>
<td>§ &lt;FieldName&gt;,&lt;FieldType&gt;,&lt;FieldLength&gt;,&lt;FieldValue&gt; ‡ Data message. &lt;FieldName&gt; is the name specified in the command. &lt;FieldType&gt; is the type of data in the field: A (alphanumeric), N (numeric), D (date), T (time), or $ (currency). &lt;FieldLength&gt; is the number of characters in the field. &lt;FieldValue&gt; is the value in the current customer record.</td>
</tr>
</tbody>
</table>

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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)
Commands and notification events

See also

- AGTListDataFields on page 156
- AGTUpdateField on page 242
AGTReadyNextItem

AGTReadyNextItem is a working command.

Alternate name

ReadyNextItem

Function

Permits Avaya 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 on page 91.

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

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28885</td>
<td>Not attached to a job. Attach a job, become available for work, and retry.</td>
</tr>
<tr>
<td>E28896</td>
<td>Headset must be active. The agent's headset is disconnected. Reconnect the</td>
</tr>
<tr>
<td></td>
<td>headset and retry.</td>
</tr>
<tr>
<td>E28901</td>
<td>Not available for work. Execute AGTAvailWork and retry.</td>
</tr>
<tr>
<td>E28902</td>
<td>Already on a customer record. Release the current record with</td>
</tr>
<tr>
<td></td>
<td>AGTReleaseLine or AGTFinishedItem and retry.</td>
</tr>
</tbody>
</table>
Commands and notification events

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28903</td>
<td>Already set ready for next customer record. AGTReadyNextItem already executed.</td>
</tr>
<tr>
<td>E28904</td>
<td>Request for no further work pending. Command is disabled.</td>
</tr>
<tr>
<td>E28905</td>
<td>Request to transfer to another job is active. Retry after job transfer is complete.</td>
</tr>
<tr>
<td>E28964</td>
<td>Agent phone is busy. Release phone line before proceeding.</td>
</tr>
<tr>
<td>E28965</td>
<td>Softdialer link is down. Please wait until the link is up.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

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 terminator(ETX), ASCII x03 (socket) or carriage return (tty)
† = Message continues(ETB), ASCII x03
§ = Message data separator, ASCII x1E (socket) or comma (tty)

See also

- [AGTAvailWork](#) on page 91
- [AGTFinishedItem](#) on page 119
- [AGTManagedCall](#) on page 185
- [AGTNoFurtherWork](#) on page 197
- [AGTPreviewRecord](#) on page 199
- [AGTRleaseLine](#) on page 209
AGTRReceiveMessage

AGTRReceiveMessage is a notification event message from the server.

Alternate name

ReceiveMessage

Function

Forwards a text message sent from the Avaya 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 Avaya Proactive Contact internal error occurs.

Format

AGTRReceiveMessage N <Agentserver> <ProclID> <InvokeID> 3§
0 § M00001 § <Message>‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

| Message | The text of the message from the supervisor. The message may be up to 79 characters long. Embedded spaces and special characters are allowed. |

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28900</td>
<td>§ &lt;message&gt;‡ An Avaya Proactive Contact system internal error occurred. Error message text follows.</td>
</tr>
</tbody>
</table>
Commands and notification events

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>M00001</td>
<td>§ &lt;Message&gt;‡ Data message. The text of the message the supervisor sent.</td>
</tr>
</tbody>
</table>

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

See also

AGTSendMessage on page 214
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.

Avaya Proactive Contact expects to receive at most one of these parameters, not both. If the agent application provides both parameters, Avaya Proactive Contact uses the <ScriptLabel> parameter but not the <MessageNo> parameter.

- If the agent application provides the <ScriptLabel> parameter, Avaya Proactive Contact executes the matching label in the telephny.spt script file and releases the line.

- If the agent application provides the <MessageNo> parameter, Avaya 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, Avaya 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‡
Commands and notification events

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

Data parameters

<table>
<thead>
<tr>
<th>ScriptLabel</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageNo</td>
<td>Optional parameter. Value is the message number to play to the customer. These numbers must appear in the voicemsg.cfg file on the Avaya Proactive Contact system. Use up to three digits, ranging from 1 to 255.</td>
</tr>
</tbody>
</table>

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28866</td>
<td>Failed to release the line for the Managed Dialing preview (no telephone line available). This message results when Avaya Proactive Contact is in the process of placing the preview call.</td>
</tr>
<tr>
<td>E28885</td>
<td>Not attached to a job. Retry when attached to a job, available for work, and working with a customer record.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S28833</td>
<td>Pending.</td>
</tr>
</tbody>
</table>

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†
AGTRleaseLine R Agent server 6111 47 2§
0 §
M00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- AGTFinishedItem on page 119
- AGTListKeys on page 161
AGTReserveHeadset

AGTReserveHeadset is a connection command.

Alternate name

ReserveHeadset

Function

Identifies a headset and reserves memory for a headset connection. Avaya 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 x03
‡ = 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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28850</td>
<td>Internal Avaya Proactive Contact system error: cannot open channel to operator monitor process. Unable to reserve headset ID. May indicate an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>E28870</td>
<td>There is already a request to reserve the headset ID pending.</td>
</tr>
<tr>
<td>AGTReserveHeadset</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td>E28871</td>
<td>$&lt;\text{HeadsetID}&gt;$ $&lt;\text{Message}&gt;$† Unexpected return message from Avaya Proactive Contact. Message text follows headset ID.</td>
</tr>
<tr>
<td>E28922</td>
<td>No reserve headset ID request pending. May indicate an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>E28923</td>
<td>$&lt;\text{HeadsetID}&gt;$† The requested headset ID is reserved.</td>
</tr>
<tr>
<td>E50611</td>
<td>$&lt;\text{HeadsetID}&gt;$† The requested headset ID is in use.</td>
</tr>
<tr>
<td>E50612</td>
<td>$&lt;\text{HeadsetID}&gt;$† No more headsets are permitted on the system.</td>
</tr>
<tr>
<td>E50613</td>
<td>$&lt;\text{HeadsetID}&gt;$† Failure to access the headset ID table. May indicate a permissions problem for the headset ID table file on the Avaya Proactive Contact system, a corrupt headset ID table file, or an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
<tr>
<td>S28833</td>
<td>Pending.</td>
</tr>
</tbody>
</table>

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

### See also

- [AGTConnHeadset](#) on page 100
- [AGTFreeHeadset](#) on page 121
AGTSendMessage

AGTSendMessage is a system command.

Alternate name

SendMessage

Function

Sends a message to the Avaya 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> <ProcID> <InvokeID> 1§
<Message>‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = 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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

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 $
Μ00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also
AGTReceiveMessage on page 207
AGTSetCallback

AGTSetCallback is a working command.

Alternate name

SetCallback

Function

Sets the recall time for a customer record. Avaya 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 Avaya Proactive Contact system time format.

It also lists the number of telephone numbers in the customer record.

The recall time format may be an absolute time or a time between the current time and the time when the recall should occur (an elapsed time).

If Avaya Proactive Contact configuration uses an elapsed time format, the vlocale.cfg file may 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.

Avaya 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> <ProclID> <InvokeID> 5§
<Date> § <Time> § <PhoneIndex> § <RecallName> §
<RecallNumber.§

AGTSetCallback C <Client> <ProclID> <InvokeID> 3§
<Date> § <Time> § <PhoneIndex.§

AGTSetCallback C <Client> <ProclID> <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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Date</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>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</td>
</tr>
<tr>
<td><strong>PhoneIndex</strong></td>
<td>Index of the Avaya 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.</td>
</tr>
<tr>
<td><strong>RecallName</strong></td>
<td>This is the customer name field to contact during the recall. There may 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.</td>
</tr>
<tr>
<td><strong>RecallNumber</strong></td>
<td>This is the telephone number to use for the recall. There may 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 Avaya Proactive Contact system. When RecallNumber exits, Avaya Proactive Contact ignores the PhoneIndex entry.</td>
</tr>
</tbody>
</table>

*Avaya Proactive Contact systems use the 10-character date format with 4-digit years, for example 1998/11/20.*
## Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28800</td>
<td>§ &lt;Minutes&gt;‡Recall cannot be less than &lt;Minutes&gt; from current time. This value is from the RECALLLIMIT settings in the vlocale.cfg file on the Avaya Proactive Contact system.</td>
</tr>
<tr>
<td>E28831</td>
<td>Date field has non-numeric value. Retry the command with the &lt;Date&gt; parameter set to a numeric value.</td>
</tr>
<tr>
<td>E28832</td>
<td>§&lt;Input&gt; § &lt;DateFormat&gt;‡ &lt;Input&gt; used as the date parameter does not match the Avaya Proactive Contact system &lt;DateFormat&gt; shown here. Retry command with correct date format.</td>
</tr>
<tr>
<td>E28833</td>
<td>Invalid month in date. Retry with month in the range 01 to 12.</td>
</tr>
<tr>
<td>E28834</td>
<td>Invalid year in date. Retry with year set to current year or current year plus some number.</td>
</tr>
<tr>
<td>E28835</td>
<td>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.</td>
</tr>
<tr>
<td>E28836</td>
<td>Invalid format character found. Check date for slash (/) versus backslash () and time for colons or other extra characters.</td>
</tr>
<tr>
<td>E28837</td>
<td>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.</td>
</tr>
<tr>
<td>E28838</td>
<td>Invalid minute in time. Minute value must fall in the range 00 to 59.</td>
</tr>
<tr>
<td>E28839</td>
<td>Invalid second in time. Retry command without any seconds value in the time parameter.</td>
</tr>
<tr>
<td>E28840</td>
<td>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.</td>
</tr>
<tr>
<td>E28841</td>
<td>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.</td>
</tr>
<tr>
<td>E28842</td>
<td>, &lt;status&gt; Invalid recall telephone selected. The telephone number specified by the &lt;index&gt; parameter is not valid. The &lt;status&gt; 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.</td>
</tr>
<tr>
<td>E28843</td>
<td>§ &lt;PhoneNumber&gt;‡The &lt;PhoneNumber&gt; is invalid. Either the format of the number does not match the standard phone format on Avaya Proactive Contact or the telephone number itself is not valid.</td>
</tr>
</tbody>
</table>
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 Avaya 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, Avaya 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. Avaya 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)
Commands and notification events

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

See also

- [AGTFinishedItem](#) on page 119
- [AGTListCallbackFmt](#) on page 149
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.

Avaya 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> <ProcID> <InvokeID> 2$<ListType> § <FieldName>
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)
Commands and notification events

Data parameters

<table>
<thead>
<tr>
<th>ListType</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>A field name from the calling list, as shown by AGTListDataFields. Up to 19 characters, alphanumeric, case sensitive, no embedded spaces.</td>
</tr>
</tbody>
</table>

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28885</td>
<td>Not attached to a job. Attach a job with AGTAttachJob and retry.</td>
</tr>
<tr>
<td>E28891</td>
<td>Invalid &lt;ListType&gt; parameter. Specify I (inbound) or O (outbound) calling list type.</td>
</tr>
<tr>
<td>E28892</td>
<td>No inbound calling list fields available. Retry using &lt;ListType&gt; O.</td>
</tr>
<tr>
<td>E28893</td>
<td>No outbound calling list fields available. Retry using &lt;ListType&gt; I.</td>
</tr>
<tr>
<td>E28894</td>
<td>The &lt;FieldName&gt; parameter does not match any of the field names in the calling list for the attached job. Execute AGTListDataFields to get correct field names.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

Examples

Agent Application to Agent Binary

AGTSetDataField C COriginator_ID 11111 178 2§
O §
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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)
See also

- AGTCallNotify on page 94
- AGTClearDataSet on page 98
- AGTDetachJob on page 102
- AGTListDataFields on page 156
- AGTPreviewRecord on page 199
- AGTSetNotifyKeyField on page 224
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> <ProcID> <InvokeID> 2§ <ListType> § <FieldName> ‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

<table>
<thead>
<tr>
<th>ListType</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>A field name from the calling list, as shown by AGTListDataFields. Up to 19 characters, alphanumeric, case sensitive, no embedded spaces.</td>
</tr>
</tbody>
</table>
AGTSetNotifyKeyField

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28885</td>
<td>Not attached to a job. Execute AGTAttachJob and retry the command.</td>
</tr>
<tr>
<td>E28891</td>
<td>Must specify either inbound or outbound operation. Retry the command with the &lt;ListType&gt; parameter set to I or O.</td>
</tr>
<tr>
<td>E28892</td>
<td>No inbound calling list fields available. Retry with &lt;ListType&gt; O.</td>
</tr>
<tr>
<td>E28893</td>
<td>No outbound calling list fields available. Retry with &lt;ListType&gt; I.</td>
</tr>
<tr>
<td>E28894</td>
<td>&lt;FieldName&gt; not found. The &lt;FieldName&gt; 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.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

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

See also

- AGTAttachJob on page 88
- AGTCallNotify on page 94
- AGTDetachJob on page 102
- AGTListDataFields on page 156
AGTSetPassword

AGTSetPassword is a system command.

Alternate name

SetPassword

Function

Sets the agent’s password on Avaya Proactive Contact. Enables agents to change their password.

Availability

This command is available when the agent is logged in to Avaya Proactive Contact.

Format

AGTSetPassword <UserID> <PresentPW> <NewPW>

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

<table>
<thead>
<tr>
<th>UserID</th>
<th>Agent’s user ID.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PresentPW</td>
<td>Agent’s current password.</td>
</tr>
<tr>
<td>NewPW</td>
<td>Agent’s new password.</td>
</tr>
</tbody>
</table>

Password Rules

An Avaya 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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E70013</td>
<td>Avaya Proactive Contact is not configured to allow an agent to change his or her password</td>
</tr>
<tr>
<td>E70014</td>
<td>Avaya Proactive Contact is configured with a locked password file. The agent cannot change his or her password.</td>
</tr>
<tr>
<td>E70015</td>
<td>The agent does not have access privileges to set a password. Contact your system administrator to reset a password.</td>
</tr>
<tr>
<td>E70016</td>
<td>The password the agent entered is not correct. Retry using a different password that follows the password rules.</td>
</tr>
<tr>
<td>E70017</td>
<td>The new password entry is incorrect. Retry using a different password that follows the password rules.</td>
</tr>
</tbody>
</table>
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. Avaya Proactive
Contact routes calls to an agent based on the Unit ID the agent selected at login.
Each execution of AGTSetUnit resets the Unit ID value.
AGTDetachJob clears the Unit ID value setting.
The following example illustrates how Avaya 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. Avaya 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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)
Data parameters

| UnitID | The Unit ID value selected by the agent. |

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28885</td>
<td>Not attached to a job; attach a Unit Work List job and retry.</td>
</tr>
<tr>
<td>E28886</td>
<td>The attached job is not a Unit Work List job. Detach the current job,</td>
</tr>
<tr>
<td></td>
<td>attach a Unit Work List job, and retry.</td>
</tr>
<tr>
<td>E28887</td>
<td>Already available for work; cannot change Unit ID value. Execute the</td>
</tr>
<tr>
<td></td>
<td>AGTNoFurtherWork command and retry.</td>
</tr>
<tr>
<td>E28888</td>
<td>The specified Unit ID value is not a Unit ID on the job. If necessary, use</td>
</tr>
<tr>
<td></td>
<td>AGTListUnits command to list the units for the attached job. Select a</td>
</tr>
<tr>
<td></td>
<td>valid Unit ID value and retry.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

Examples

Agent Application to Agent Binary

AGTSendMessage C COriginator_ID 11111 1 1§
HELP ME†

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‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- AGTAttachJob on page 88
- AGTDetachJob on page 102
- AGTListUnits on page 166
Commands and notification events

- AGTNoFurtherWork on page 197
AGTSetWorkClass

AGTSetWorkClass is a system command.

Alternate name

SetWorkClass

Function

Transmits the agent type to Avaya Proactive Contact.

Avaya 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.

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Job Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>inbound, blend</td>
</tr>
<tr>
<td>Outbound</td>
<td>outbound, blend, Sales Verification, Unit Work List</td>
</tr>
<tr>
<td>Blend¹</td>
<td>blend</td>
</tr>
<tr>
<td>Managed</td>
<td>Managed Dialing</td>
</tr>
<tr>
<td>Person to Person</td>
<td>outbound, inbound, blend</td>
</tr>
</tbody>
</table>

¹. 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.

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.
Commands and notification events

Format

\[
\text{AGTSetWorkClass C} \ <\text{Client}> \ <\text{ProcID}> \ <\text{InvokeID}> \ 1\§ \\
\ <\text{ClassID}>\‡
\]

\(\§ = \text{Message data separator, ASCII x1E (socket) or comma (tty)}\)

\(\‡ = \text{Message continues(ETB), ASCII x03}\)

\(\‡ = \text{Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)}\)

Data parameters

<table>
<thead>
<tr>
<th>ClassID</th>
<th>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).</th>
</tr>
</thead>
</table>

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28882</td>
<td>The agent is available for work; cannot change class. Execute AGTNoFurtherWork and retry.</td>
</tr>
<tr>
<td>E28883</td>
<td>Invalid agent type. Retry with &lt;ClassID&gt; set to I, O, B, P, or M.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

Examples

**Agent Application to Agent Binary**

\[
\text{AGTSetWorkClass C} \ \text{CO}\text{Originator_ID 11111 168 1}\§ \\
\ \text{O}‡
\]

**Agent Binary to Agent Application**

\[
\text{AGTSetWorkClass R} \ \text{Agent server 8497 168 2}§ \\
\ \text{0}\
\ \text{M00000}‡
\]

\(\§ = \text{Message data separator, ASCII x1E (socket) or comma (tty)}\)

\(\‡ = \text{Message continues(ETB), ASCII x03}\)

\(\‡ = \text{Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)}\)

See also

- AGTAvailWork on page 91
- AGTNoFurtherWork on page 197
AGTSetWorkClass

- AGTLogonAcd on page 182
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 Avaya Proactive Contact system has been established.

The client must respond with the AGTLogon command to start a session.

Format

AGTSTART N <Agentserver> <ProclD> <InvokeID> 2§ 0 § AGENT_STARTUP‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None.

Return values

None.

Examples

Agent Binary to Agent Application

AGTSTART N Agent server 17970 0 2§ 0 § AGENT_STARTUP‡

Code | Description
--- | ---
E28858 | `<AgentName>` The `<AgentName>` logon exceeds the Avaya Proactive Contact system limit on the number of agents. This error can occur if client sessions for the agent application are left running on Avaya Proactive Contact after an agent workstation crashes.
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

AGTLogon on page 180
AGTSSystemError

AGTSSystemError is a notification event message from the server.

Alternate name

SystemError

Function

Notifies the agent application that Avaya Proactive Contact has detected an internal error.

If the agent binary receives a message from Avaya Proactive Contact about the error, the agent binary terminates.

If the error is an I/O error between Avaya Proactive Contact and the agent binary, the agent binary continues. Consult the error message and Avaya Proactive Contact logs to determine the problem.

Format

AGTSSystemError N <Agentserver> <ProclID> <InvokeID> 2§
1 § <ErrorMessage>‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

| ErrorMessage | The Avaya Proactive Contact error message number and any data or information that is part of that message. |

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28859</td>
<td>\agent, &lt;AgentName&gt;, &lt;slot_number&gt; The agent logon is invalid. The first value is always agent, followed by the agent user name and the operator slot number Avaya Proactive Contact assigned to the agent name. Indicates an Avaya Proactive Contact system problem.</td>
</tr>
</tbody>
</table>
### 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 x03

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

### See also

- [AGTDumpData](#) on page 111
- [AGTLogonStart](#) on page 168
AGTTransferCall

AGTTransferCall is a working command.

Alternate name

TransferCall

Function

Transfers a customer call on Avaya 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 Avaya 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 x03
‡ = 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 Avaya Proactive Contact system. |

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28866</td>
<td>Telephone line is not available. Avaya Proactive Contact cannot place the call. Retry later.</td>
</tr>
<tr>
<td>E28867</td>
<td>Telephone line is not off-hook. Retry the command with an active telephone connection to the customer.</td>
</tr>
<tr>
<td>E29950</td>
<td>This feature is not available on an Avaya Proactive Contact with CTI system.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

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

See also

- [AGTHookflashLine](#) on page 133
- [AGTManualCall](#) on page 187
- [AGTMoFlashBlind](#) on page 189
- [AGTMoFlashSupv](#) on page 193
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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None.

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28866</td>
<td>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.</td>
</tr>
<tr>
<td>E28867</td>
<td>Telephone line is not off-hook. There is no call on hold. Customer may have been cut off rather than placed on hold.</td>
</tr>
<tr>
<td>E29950</td>
<td>This feature is not available on an Avaya Proactive Contact with CTI system.</td>
</tr>
</tbody>
</table>
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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

AGTHoldCall on page 131
AGTUpdateField

AGTUpdateField is a working command.

Alternate name

UpdateField

Function

Use this command to update the calling list during calling activities. The application may 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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

<table>
<thead>
<tr>
<th>ListType</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>A field name from the calling list, as shown by AGTListDataFields. Up to 19 alphanumeric characters, case sensitive, no embedded spaces.</td>
</tr>
<tr>
<td>NewValue</td>
<td>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).</td>
</tr>
</tbody>
</table>
Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E28885</td>
<td>Not attached to a job. Retry when attached to a job, available for work, and working with a customer record.</td>
</tr>
<tr>
<td>E28891</td>
<td>Must specify inbound or outbound operation. Retry with the &lt;ListType&gt; parameter set to either I or O.</td>
</tr>
<tr>
<td>E28892</td>
<td>No inbound calling list fields available. Retry with &lt;ListType&gt; O.</td>
</tr>
<tr>
<td>E28893</td>
<td>No outbound calling list fields available. Retry with &lt;ListType&gt; I.</td>
</tr>
<tr>
<td>E28894</td>
<td>&lt;FieldName&gt; not found. The system did not find the field name specified in the calling list. Field names are case sensitive and may contain underscores (_) in place of spaces.</td>
</tr>
<tr>
<td>E28912</td>
<td>Customer record is not available for update. The agent is not working with a customer record.</td>
</tr>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

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$O §M00000‡

§ = Message data separator, ASCII x1E (socket) or comma (tty)
‡ = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

See also

- [AGTListDataFields](#) on page 156
- [AGTReadField](#) on page 202
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 x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

See also

- AGTAutoReleaseLine on page 90
- AGTXferTrunkHangup on page 245
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> <ProclID> <InvokeID> 2§
0 § M00000‡
§ = Message data separator, ASCII x1E (socket) or comma (tty)
† = Message continues(ETB), ASCII x03
‡ = Message terminator(ETX), ASCII x03 (socket) or carriage return (tty)

Data parameters

None

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

See also

- AGTAutoReleaseLine on page 90
- AGTXferCustHangup on page 244
Commands and notification events
Appendix A: Completion codes

The following table shows the completion codes for an Avaya Proactive Contact.

**Note:**
The values 20-34 and 51-85 are available for agent completion codes.

<table>
<thead>
<tr>
<th>Code</th>
<th>Keyword</th>
<th>Type</th>
<th>Description</th>
<th>Report Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>NOTCALLED</td>
<td>System</td>
<td>The account has not been called.</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>CODE1</td>
<td>System</td>
<td>Reserved for the system.</td>
<td></td>
</tr>
<tr>
<td>002</td>
<td>ERROR</td>
<td>System</td>
<td>The system detected an invalid phone number.</td>
<td></td>
</tr>
<tr>
<td>003</td>
<td>TIMEOUT</td>
<td>System</td>
<td>The system did not receive a dial tone.</td>
<td>Timed out</td>
</tr>
<tr>
<td>004</td>
<td>HANG_PORT</td>
<td>System</td>
<td>The line was idle after the system dialed the customer record phone number.</td>
<td></td>
</tr>
<tr>
<td>005</td>
<td>NOTINZONE</td>
<td>System</td>
<td>The local time for the customer phone is outside calling hours.</td>
<td>Not within legal hours</td>
</tr>
<tr>
<td>006</td>
<td>MOFLASH_B</td>
<td>Agent</td>
<td>Used for native voice and data transfer. An agent transfers a call to an inbound agent without remaining on the line.</td>
<td>Blind transfer</td>
</tr>
<tr>
<td>007</td>
<td>HANG_TRANS</td>
<td>System</td>
<td>No agent is available for a supervisor transfer.</td>
<td></td>
</tr>
<tr>
<td>008</td>
<td>TDSS_HF_B</td>
<td>Agent</td>
<td>ADAPTS API: the agent transfers a call without remaining on the call. This is known as blind hook flash transfer.</td>
<td></td>
</tr>
<tr>
<td>009</td>
<td></td>
<td>System</td>
<td>Reserved for the system.</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td></td>
<td>System</td>
<td>Reserved for the system.</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>BUSY</td>
<td>System</td>
<td>The system detected a busy signal.</td>
<td></td>
</tr>
<tr>
<td>012</td>
<td>CONTTONE</td>
<td>System</td>
<td>The system detected a continuous tone, such as a fax or modem.</td>
<td></td>
</tr>
<tr>
<td>013</td>
<td>AUTOVOICE</td>
<td>System</td>
<td>The system detected an answering machine.</td>
<td></td>
</tr>
</tbody>
</table>
## Completion codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Keyword</th>
<th>Type</th>
<th>Description</th>
<th>Report Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>014</td>
<td>VOICE</td>
<td>System</td>
<td>Interim code when a person is on the line.</td>
<td></td>
</tr>
<tr>
<td>015</td>
<td>NOANSWER</td>
<td>System</td>
<td>The call placed was not answered.</td>
<td></td>
</tr>
<tr>
<td>016</td>
<td>RINGING</td>
<td>Agent</td>
<td>Can be user defined but is usually defined as a phone call that was still ringing but was passed to an agent.</td>
<td></td>
</tr>
<tr>
<td>017</td>
<td>CUSTHU</td>
<td>Agent</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>018</td>
<td>TRANSFER</td>
<td>Agent</td>
<td>Can be user defined but is usually defined as a transfer release.</td>
<td>Transferred</td>
</tr>
<tr>
<td>019</td>
<td>RECALL</td>
<td>Agent</td>
<td>Can be user defined but is usually defined as a recall release.</td>
<td></td>
</tr>
<tr>
<td>020-034</td>
<td>Agent</td>
<td></td>
<td>Customer assigned codes used by agents.</td>
<td></td>
</tr>
<tr>
<td>035</td>
<td>CANCEL</td>
<td>System</td>
<td>Can be user defined but is usually defined as the agent cancelled the managed call.</td>
<td></td>
</tr>
<tr>
<td>036</td>
<td>INTERCEPT</td>
<td>System</td>
<td>Special Information Tone (SIT) received that indicates an operator intercepted the call.</td>
<td></td>
</tr>
<tr>
<td>037</td>
<td>NOCIRCUIT</td>
<td>System</td>
<td>SIT received that indicates the circuits were unavailable.</td>
<td></td>
</tr>
<tr>
<td>038</td>
<td>DISCONN</td>
<td>System</td>
<td>SIT received that indicates the call was a disconnected number.</td>
<td></td>
</tr>
<tr>
<td>039</td>
<td>VACANT</td>
<td>System</td>
<td>SIT received that indicates the call cannot be completed as dialed.</td>
<td></td>
</tr>
<tr>
<td>040</td>
<td>REORDER</td>
<td>System</td>
<td>The call resulted in a fast busy tone.</td>
<td></td>
</tr>
<tr>
<td>041</td>
<td>R_RINGING</td>
<td>System</td>
<td>Reserved.</td>
<td></td>
</tr>
<tr>
<td>042</td>
<td>LINEFAIL</td>
<td>System</td>
<td>A failure on the phone line occurred.</td>
<td></td>
</tr>
<tr>
<td>043</td>
<td>OP_RECALL</td>
<td>System</td>
<td>Operator set recall.</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Keyword</td>
<td>Type</td>
<td>Description</td>
<td>Report Header</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>----------</td>
<td>-------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>044</td>
<td>DTMF_V</td>
<td>System</td>
<td>DTMF tone detected.</td>
<td>Voice DTMF</td>
</tr>
<tr>
<td>045</td>
<td>HU_INB</td>
<td>System</td>
<td>The customer hung up while in the inbound wait queue.</td>
<td></td>
</tr>
<tr>
<td>046</td>
<td>HU_OUT</td>
<td>System</td>
<td>The customer hung up while in the outbound wait queue.</td>
<td></td>
</tr>
<tr>
<td>047</td>
<td>HANG_INB</td>
<td>System</td>
<td>An agent was unavailable for the inbound call.</td>
<td></td>
</tr>
<tr>
<td>048</td>
<td>HANG_OUT</td>
<td>System</td>
<td>An agent was unavailable for the outbound call.</td>
<td></td>
</tr>
<tr>
<td>049</td>
<td>OPDIED</td>
<td>System</td>
<td>The agent session ended abnormally.</td>
<td></td>
</tr>
<tr>
<td>050</td>
<td>R_HSONHOOK</td>
<td>System</td>
<td>The agent headset disconnected from Avaya Proactive Contact.</td>
<td></td>
</tr>
<tr>
<td>051-088</td>
<td></td>
<td>Agent</td>
<td>Customer assigned codes used by agents.</td>
<td></td>
</tr>
<tr>
<td>089</td>
<td>MANAGEDA</td>
<td>Agent</td>
<td>Managed Dial: Managed non-connection A.</td>
<td></td>
</tr>
<tr>
<td>090</td>
<td>MANAGEDB</td>
<td>Agent</td>
<td>Managed Dial: Managed non-connection B.</td>
<td></td>
</tr>
<tr>
<td>091</td>
<td>VIRTVOICE</td>
<td>System</td>
<td>Virtual Agent: Virtual message to VOICE, a person.</td>
<td></td>
</tr>
<tr>
<td>092</td>
<td>VIRTUALAV</td>
<td>System</td>
<td>Virtual Agent: Virtual message to AUTOVOICE, a calling machine.</td>
<td></td>
</tr>
<tr>
<td>093</td>
<td>SOLD</td>
<td>Agent</td>
<td>Sales Verification: Sold campaign.</td>
<td></td>
</tr>
<tr>
<td>094</td>
<td>VERIFIED</td>
<td>Agent</td>
<td>Sales Verification: Sale verified.</td>
<td>Verified sale</td>
</tr>
<tr>
<td>095</td>
<td>UNVERIFIED</td>
<td>Agent</td>
<td>Sales Verification: Sale not verified.</td>
<td></td>
</tr>
<tr>
<td>096-097</td>
<td></td>
<td>System</td>
<td>Reserved for the system.</td>
<td></td>
</tr>
<tr>
<td>098</td>
<td>AORECALL</td>
<td>Agent</td>
<td>Agent Owned Recall.</td>
<td></td>
</tr>
<tr>
<td>099</td>
<td></td>
<td>System</td>
<td>Reserved for the system.</td>
<td></td>
</tr>
<tr>
<td>100-200</td>
<td></td>
<td>Agent</td>
<td>Customer assigned</td>
<td></td>
</tr>
</tbody>
</table>
Completion codes
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 Avaya Proactive Contact Agent API (Agent API). Record formats appear in alphabetical order by type.

This section contains the following topics:

- **Command message formats** on page 251
- **Response message formats** on page 253
- **Data message formats** on page 253
- **Notification event message formats** on page 255

### 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‡`
- `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‡`
- `AGTFinishedItem C <Client> <ProcID> <InvokeID> 1 §<CompCode>‡`
- `AGTFreeHeadset C <Client> <ProcID> <InvokeID> 0‡`
Agent API message format quick reference

AGTGetHeadsetVol C <Client> <ProcID> <InvokeID> 0‡
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‡
AGTLogoffAcd C <Client> <ProcId> <InvokeID> 0‡
AGTLogon C <Client> <ProcID> <InvokeID> 2 §<AgentName> §<Password>‡
AGTLogonAcd 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‡
Response message formats

All commands receive the generic complete message (Avaya Proactive Contact code M00000) when the action is complete. Many commands receive the generic pending message (Avaya Proactive Contact code S28833) that indicates a command requires processing by Avaya 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>§
Agent API message format quick reference

Mouth>

AGTListCallbackFmt D Agent <ProcID> <InvokeID> 4 $0$
server M00001$
<Format>§
<Phones>‡

AGTListCallFields D Agent <ProcID> <InvokeID> <n> $0$
server M00001$
<FieldName>,
<FieldLength>,
 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>…‡
AGTManagedCall D Agent <ProcID> <InvokeID> 3 $0$ server M00001§<CallStatus>‡
AGTReadField D Agent <ProcID> <InvokeID> 3 $0$ server M00001§<FieldName>,<FieldType>,<FieldLength>,<FieldValue> ‡

Notification event message formats

AGTAutoReleaseLine N <Agentserver> <ProcID> <InvokeID> 2 $0$ server M00000‡
AGTCallNotify N Agent <ProcID> <InvokeID> 5 0 server § M00001 $<OpMesg>[^<WaitMsg>] $<CallType> $<NotifyFieldName>, $<NotifyFieldData> ‡
AGTCallNotify N Agent <ProcID> <InvokeID> <n> $0$ server § M00001 $<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‡
AGTicbFeNotif N Agent <ProcID> <InvokeID> 2 §0§ server S28972‡

AGTicbOffline N Agent <ProcID> <InvokeID> 2 §0§ server M00000‡

AGTicbOnline 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>‡

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>‡§

AGTXferCustHangup N <Agentserver> <ProcID> <InvokeID> 2§ 0 § M00000‡

AGTXferTrunkHangup N <Agentserver> <ProcID> <InvokeID> 2§ 0 § M00000‡
Appendix C: System messages

The purpose of this section is to provide a table of system messages that may be returned to the Avaya Agent API (Agent API).

This section contains the following topics:

- Error messages on page 257
- Data messages on page 295
- Pending messages on page 297

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 may 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 Avaya Proactive Contact system problem. For assistance with these errors, contact the Avaya Proactive Contact system administrator.

Note:

Avaya Proactive Contact may 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.

E00518,agent, <ListName>

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTListCallFields</td>
<td>There is no calling list with &lt;ListName&gt; on the system. The agent entry is the name of the process on the Avaya Proactive Contact system returning the error. &lt;ListName&gt; is the name submitted by the client process.</td>
</tr>
<tr>
<td>AGTListKeys</td>
<td>Unable to open the keys file. Indicates an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>Error text</td>
<td>E00518: Calling list %s does not exist.</td>
</tr>
</tbody>
</table>
E28800 § <minutes>‡

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>Recall cannot be less than &lt;minutes&gt; from current time. This value is from the RECALLLIMIT settings in the vlocale.cfg file on the Avaya Proactive Contact system.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28800: Recall cannot be less than %s minutes.</td>
</tr>
</tbody>
</table>

E28804

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAttachJob</td>
<td>Job &lt;JobName&gt; is not running, not active.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28804: Job %s is not running.</td>
</tr>
</tbody>
</table>

E28805 § <JobName> ‡

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>&lt;JobName&gt; is not ready. The attached job is active, but is not placing calls. Retry the login after job starts placing calls.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28805: Job %s is not ready. Join the job later.</td>
</tr>
</tbody>
</table>

E28812

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogon</td>
<td>&lt;AgentName&gt; is logged in to another application process. The agent may be logged in at another location. Retry the login with a different &lt;AgentName&gt;.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28812: Agent %s already logged on. Access is denied.</td>
</tr>
</tbody>
</table>
### E28813

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>No more agents of this type may join the job. Job parameters include a maximum number of agents of each type (work class) that may log into the job. See AGTSetWorkClass on page 231.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28813: Maximum %s agent limit reached.</td>
</tr>
</tbody>
</table>

### E28814

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>Managed agents may 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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28814: Managed agents cannot join this job.</td>
</tr>
</tbody>
</table>

### E28815

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28815: Sales verification with unit work lists is not permitted.</td>
</tr>
</tbody>
</table>

### E28816

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>Only inbound agents may join this job. The agent application agent type (work class) does not match the job type. Set the agent type to I and retry.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28816: Only inbound agents are permitted.</td>
</tr>
</tbody>
</table>
### E28817

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>Only outbound agents may join this job. The agent application agent type (work class) does not match the job type. Set the agent type to O and retry.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28817: Only outbound agents are permitted.</td>
</tr>
</tbody>
</table>

### E28818

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>Only Outbound or Managed agents may 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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28818: Only outbound or Managed agents are permitted.</td>
</tr>
</tbody>
</table>

### E28819

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>Only Outbound agents may 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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28819: Only outbound agents are permitted on a Sales Verification job.</td>
</tr>
</tbody>
</table>

### E28831

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>Date field has non-numeric value. Retry the command with the &lt;Date&gt; parameter set to a numeric value.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28831: Field has non-numeric value.</td>
</tr>
</tbody>
</table>
### E28832 § <input>§ <DateFormat> §

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>&lt;input&gt; used as the date parameter does not match the Avaya Proactive Contact &lt;DateFormat&gt; shown here. Retry command with correct date format.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28832: Date ccyy/mm/dd %s submitted doesn’t match %s format.</td>
</tr>
</tbody>
</table>

### E28833

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>Invalid month in date. Retry with month in the range 01 to 12.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28833: Date has an invalid month.</td>
</tr>
</tbody>
</table>

### E28834

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>Invalid year in date. Retry with year set to current year or current year plus some number.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28834: Date has an invalid year.</td>
</tr>
</tbody>
</table>

### E28835

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28835: Date has an invalid day.</td>
</tr>
</tbody>
</table>
### E28836

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>Invalid format character found. Check date for slash (/) versus backslash () and time for colons or other extra characters.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28836: Invalid format character found.</td>
</tr>
</tbody>
</table>

### E28837

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28837: Time has an invalid hour.</td>
</tr>
</tbody>
</table>

### E28838

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>Invalid minute in time. Minute value must fall in the range 00 to 59.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28838: Time has an invalid minute.</td>
</tr>
</tbody>
</table>

### E28839

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>Invalid second in time. Retry command without any seconds value in the time parameter.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28839: Time has an invalid second.</td>
</tr>
</tbody>
</table>
### E28840

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28840: Time is not in the correct format.</td>
</tr>
</tbody>
</table>

### E28841

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28841: Invalid phone.</td>
</tr>
</tbody>
</table>

### E28842, `<status>`

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>Invalid recall telephone selected. The telephone number specified by the <code>&lt;index&gt;</code> parameter is not valid. The <code>&lt;status&gt;</code> 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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28842: Invalid phone number. Phonestat: %s.</td>
</tr>
</tbody>
</table>

### E28843, `<PhoneNumber>`

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTManualCall</td>
<td>The <code>&lt;PhoneNumber&gt;</code> is invalid. Either the format of the number does not match the standard phone format on Avaya Proactive Contact, or the telephone number itself is not valid.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28843: Invalid phone number - %s.</td>
</tr>
</tbody>
</table>
## E28847

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>Date is before the current date. Retry the command with today’s date or some future date.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28847: Date is before the current date.</td>
</tr>
</tbody>
</table>

## E28848

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>Recall time and date outside time zone. The recall time setting is for the Avaya 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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28848: Recall time is outside the limits for the time zone.</td>
</tr>
</tbody>
</table>

## E28849, <process>

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetCallback</td>
<td>Because the time zone for the customer record is unknown, Avaya Proactive Contact cannot tell if the recall time is within legal calling times. &lt;Process&gt; discovered the bad time zone; it is always agent.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28849: %s - Time zone for the record is not known.</td>
</tr>
</tbody>
</table>
### E28850

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTConnHeadset</td>
<td>Internal Avaya Proactive Contact system error: cannot open a channel to the operator monitor process; unable to connect headset ID. This message may indicate an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>AGTReserveHeadset</td>
<td>Internal Avaya Proactive Contact system error: cannot open a channel to the operator monitor process; unable to reserve headset ID. May indicate an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>AGTAdjustHeadset</td>
<td>Cannot open the channel to the operator monitor process. Indicates an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>AGTGetHeadsetVol</td>
<td></td>
</tr>
<tr>
<td>Error text</td>
<td>E28850: ERROR: Cannot open channel to operator monitor process.</td>
</tr>
</tbody>
</table>

### E28851

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTGetHeadsetVol</td>
<td>There is no response from the operator monitor process. Indicates an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28851: No response from the operator monitor process.</td>
</tr>
</tbody>
</table>

### E28858, <AgentName> 

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSTART</td>
<td>The &lt;AgentName&gt; login exceeds the Avaya Proactive Contact limit on the number of agents. This error can occur if client sessions for the agent application are left running on the Avaya Proactive Contact system after agent workstation crashes.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28858: %s exceeded the number of agent slots available.</td>
</tr>
</tbody>
</table>
**System messages**

**E28859, <agent>, <AgentName>, <slot_number>**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSYSTEMERROR</td>
<td>The agent login is invalid. The first value is always agent, followed by the agent login name and the operator slot number Avaya Proactive Contact assigned to the agent name. Indicates an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28859: %s (%s) - Agent number returned invalid %s.</td>
</tr>
</tbody>
</table>

**E28862, <JobName>**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSYSTEMERROR</td>
<td>The agent's job selection (&lt;JobName&gt;) returned a fatal error. The agent child process supporting the client session is terminating.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28862: Avaya Proactive Contact FATAL ERROR ON SELECT %s. Process terminating.</td>
</tr>
</tbody>
</table>

**E28863**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSYSTEMERROR</td>
<td>Unknown file descriptor. Can indicate a command format problem from the agent application.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28863: ERROR. Unknown file descriptor.</td>
</tr>
</tbody>
</table>

**E28864, <MessageText>**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSYSTEMERROR</td>
<td>Unknown IPC message. Message text follows.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28864: Unknown IPC message - %s.</td>
</tr>
</tbody>
</table>
### E28865, `<Command>`

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic</td>
<td>Unknown command message. The <code>&lt;Command&gt;</code> is the text string that agent received as the command.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28865: Unknown Command message - %s.</td>
</tr>
</tbody>
</table>

### E28866

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAdjustHeadset</td>
<td>Telephone line is not available. The agent must receive a customer call to acquire an open telephone line.</td>
</tr>
<tr>
<td>AGTDialDigit</td>
<td>Telephone line is not available. The agent must receive a customer call to acquire an open telephone line.</td>
</tr>
<tr>
<td>AGTGetHeadsetVol</td>
<td>Telephone line is not available. The agent must receive a customer call to acquire an open telephone line.</td>
</tr>
<tr>
<td>AGTManualCall</td>
<td>Telephone line is not available. The agent must receive a customer call to acquire an open telephone line.</td>
</tr>
<tr>
<td>AGTFinishedItem</td>
<td>There is no line available. This error generally indicates the agent is trying to cancel a Managed Dialing call too late.</td>
</tr>
<tr>
<td>AGTHangupCall</td>
<td>A telephone line is not available. There is no call to hang up.</td>
</tr>
<tr>
<td>AGTHoldCall</td>
<td>Telephone line is not available. There is no open telephone line to place on hold.</td>
</tr>
<tr>
<td>AGTHookflashLine</td>
<td>No phone line is available. Avaya Proactive Contact does not have a telephone line available to place the transfer call.</td>
</tr>
<tr>
<td>AGTMoFlashBlind</td>
<td>No phone line is available. Avaya Proactive Contact does not have a telephone line available to place the transfer call.</td>
</tr>
<tr>
<td>AGTMoFlashSupv</td>
<td>No phone line is available. Avaya Proactive Contact does not have a telephone line available to place the transfer call.</td>
</tr>
<tr>
<td>AGTTransferCall</td>
<td>No phone line is available. Avaya Proactive Contact does not have a telephone line available to place the transfer call.</td>
</tr>
<tr>
<td>AGTReleaseLine</td>
<td>Failed to release the line for the Managed Dialing preview (no telephone line available). This message results when Avaya Proactive Contact is placing the preview call.</td>
</tr>
<tr>
<td>AGTSetCallback</td>
<td>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.</td>
</tr>
<tr>
<td>AGTUholdCall</td>
<td>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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28866: Telephone line is not available.</td>
</tr>
</tbody>
</table>
## E28867

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAdjustHeadset</td>
<td>Telephone line is not offhook. The agent has an open telephone line that is on hold.</td>
</tr>
<tr>
<td>AGTDialDigit AGTGetHeadsetVol</td>
<td>The telephone line is not offhook. Place the telephone line in offhook state and retry.</td>
</tr>
<tr>
<td>AGTHoldCall</td>
<td>Telephone line is not offhook. While there is an open telephone line, there is no call to place on hold. This message may mean the agent has already placed the call on hold.</td>
</tr>
<tr>
<td>AGTMoFlashBlind AGTMoFlashSupv AGTHookflashLine AGTTransferCall</td>
<td>Line is not offhook. There is no current telephone call to transfer.</td>
</tr>
<tr>
<td>AGTUunholdCall</td>
<td>Telephone line is not offhook. There is no call on hold. Customer may have been cut off rather than placed on hold.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28867: Telephone line is not offhook.</td>
</tr>
</tbody>
</table>

## E28868

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTListCallbackFmt</td>
<td>No recalls during inbound jobs. The current attached job is an inbound job. There is no recall format.</td>
</tr>
<tr>
<td>AGTMoFlashSupv</td>
<td>The transfer failed, try again later. There is no agent available on the specified job to take the transfer call.</td>
</tr>
<tr>
<td>AGTSetCallback</td>
<td>Recalls not permitted on inbound jobs. Avaya Proactive Contact cannot schedule recalls on inbound jobs.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28868: Recalls are not permitted on inbound calls.</td>
</tr>
</tbody>
</table>
### E28869

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAdjustHeadset</td>
<td>Headset volume must be in the range of one to eight. Retry the command using a number from one to eight in the &lt;Volume&gt; parameter.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28869: Headset volume must be set between 1 and 8.</td>
</tr>
</tbody>
</table>

### E28870

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReserveHeadset</td>
<td>There is already a request to reserve the headset ID pending.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28870: Reserve headset ID request pending.</td>
</tr>
</tbody>
</table>

### E28871 § <HeadsetID>§ <Message>‡

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReserveHeadset</td>
<td>Unexpected return message from Avaya Proactive Contact. Message text follows headset ID.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28871: Invalid headset ID - %s.</td>
</tr>
</tbody>
</table>

### E28872

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTConnHeadset</td>
<td>Headset is already connected.</td>
</tr>
<tr>
<td>AGTLogonAcd</td>
<td>The agent is already logged in to ACD.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28872: Headset is already connected.</td>
</tr>
</tbody>
</table>
System messages

**E28873**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTConnHeadset</td>
<td>Headset ID is not reserved. Execute AGTReserveHeadset and retry.</td>
</tr>
<tr>
<td>AGTDisconnHeadset</td>
<td>The headset is not reserved. There is no headset to disconnect.</td>
</tr>
<tr>
<td>AGTFreeHeadset</td>
<td>There is no headset reserved. There is nothing to free.</td>
</tr>
<tr>
<td>AGTLogonAcd</td>
<td>Headset is not reserved. Reserve a headset identification with AGTReserveHeadset before executing this command.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28873: Headset ID is not reserved nor validated.</td>
</tr>
</tbody>
</table>

**E28874**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTConnHeadset</td>
<td>A connect headset request is already pending.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28874: Connect headset request is pending.</td>
</tr>
</tbody>
</table>

**E28875**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTConnHeadset</td>
<td>The headset connect request did not register. This message may indicate an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28875: No headset connect request is pending.</td>
</tr>
</tbody>
</table>

**E28876**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTConnHeadset</td>
<td>The headset is not connected. This message may indicate that there is a problem with the telephone connection to the headset or an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>Source</td>
<td>Message Interpretation</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>AGTDisconnHeadset</td>
<td>The headset is not connected. There is no headset connection to close.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28876: Headset is not connected.</td>
</tr>
</tbody>
</table>

**E28877**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTDisconnHeadset</td>
<td>There is a disconnect headset request already pending.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28877: Disconnect headset request is pending.</td>
</tr>
</tbody>
</table>

**E28879**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTFreeHeadset</td>
<td>The headset is not disconnected. Disconnect the headset and retry.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28879: Headset is not disconnected.</td>
</tr>
</tbody>
</table>

**E28880**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTHeadsetConnBroken</td>
<td>Headset connection is broken.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28880: Headset connection is broken.</td>
</tr>
</tbody>
</table>

**E28881**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTHeadsetConnBroken</td>
<td>Headset connection re-established.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28881: Headset reconnected.</td>
</tr>
</tbody>
</table>
System messages

**E28882**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetWorkClass</td>
<td>The agent is available for work; cannot change class. Execute AGTNoFurtherWork and retry.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28882: Already available for work. Cannot change the agent type (work class).</td>
</tr>
</tbody>
</table>

**E28883**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetWorkClass</td>
<td>Invalid agent type. Retry with &lt;ClassID&gt; set to I, O, B, P, or M.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28883: Invalid agent type (work class). Type must be B, I, O, M, or P.</td>
</tr>
</tbody>
</table>

**E28884**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTListJobs</td>
<td>The agent server was unable to access Avaya Proactive Contact shared memory; usually indicates an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28884: Cannot attach to shared data memory.</td>
</tr>
</tbody>
</table>

**E28885**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>The agent application is not attached to a job. The agent must select a job before joining it.</td>
</tr>
<tr>
<td>AGTCallNotify, AGTPreviewRecord</td>
<td>The agent application is not attached to a job. The event is inappropriate.</td>
</tr>
<tr>
<td>Source</td>
<td>Message Interpretation</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AGTClearDataSet</td>
<td>The agent application is not attached to a job. Execute AGTAttachJob and retry.</td>
</tr>
<tr>
<td>AGTFinishedItem</td>
<td></td>
</tr>
<tr>
<td>AGTListCallbackFmt</td>
<td></td>
</tr>
<tr>
<td>AGTListDataFields</td>
<td></td>
</tr>
<tr>
<td>AGTListKeys</td>
<td></td>
</tr>
<tr>
<td>AGTSetDataField</td>
<td></td>
</tr>
<tr>
<td>AGTSetNotifyKeyField</td>
<td></td>
</tr>
<tr>
<td>AGTListUnits</td>
<td>Not attached to a job; attach an active Unit Work List job and retry.</td>
</tr>
<tr>
<td>AGTSetUnit</td>
<td></td>
</tr>
<tr>
<td>AGTFinishedItem</td>
<td>Not attached to a job. Retry when attached to a job, available for work, and working with a customer record.</td>
</tr>
<tr>
<td>AGTManagedCall</td>
<td></td>
</tr>
<tr>
<td>AGTReadField</td>
<td></td>
</tr>
<tr>
<td>AGTReleasedLine</td>
<td></td>
</tr>
<tr>
<td>AGTUpdateField</td>
<td></td>
</tr>
<tr>
<td>AGTReadyNextItem</td>
<td>Not attached to a job. Attach a job, become available for work, and retry.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28885: Not attached to a job.</td>
</tr>
</tbody>
</table>

**E28886**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTListUnits</td>
<td>The attached job is not a Unit Work List job; change attached job and retry.</td>
</tr>
<tr>
<td>AGTSetUnit</td>
<td></td>
</tr>
<tr>
<td>Error text</td>
<td>E28886: Unit work lists are not permitted on this job.</td>
</tr>
</tbody>
</table>

**E28887**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetUnit</td>
<td>Already available for work; cannot change unit ID value. Execute the AGTNoFurtherWork command and retry.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28887: Already available for work. Cannot change unit.</td>
</tr>
</tbody>
</table>
## System messages

### E28888

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetUnit</td>
<td>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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28888: Unit not found.</td>
</tr>
</tbody>
</table>

### E28889

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAttachJob</td>
<td>The agent application is already attached to a job. The agent must clear the current job selection before selecting a new one.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28889: Already attached to a job. Detach current job and retry.</td>
</tr>
</tbody>
</table>

### E28890

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAttachJob</td>
<td>Cannot open the job's resource file. This message usually indicates an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28890: Failure to open job % resource file.</td>
</tr>
</tbody>
</table>

### E28891

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTClearDataSet</td>
<td>Must specify inbound or outbound operation. Retry the command with the &lt;ListType&gt; parameter set to either I or O.</td>
</tr>
<tr>
<td>Source</td>
<td>Message Interpretation</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>AGTListDataFields</td>
<td>Must specify I (inbound) or O (outbound) calling list type. Retry command with &lt;ListType&gt; parameter included.</td>
</tr>
<tr>
<td>AGTReadField</td>
<td></td>
</tr>
<tr>
<td>AGTSetDataField</td>
<td></td>
</tr>
<tr>
<td>AGTSetNotifyKeyField</td>
<td></td>
</tr>
<tr>
<td>AGTUpdateField</td>
<td></td>
</tr>
<tr>
<td>Error text</td>
<td>E28891: Must specify INBOUND or OUTBOUND operation.</td>
</tr>
</tbody>
</table>

**E28892**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTClearDataSet</td>
<td></td>
</tr>
<tr>
<td>AGTListDataFields</td>
<td>There are no inbound calling list fields available. Retry with &lt;ListType&gt; O.</td>
</tr>
<tr>
<td>AGTReadField</td>
<td></td>
</tr>
<tr>
<td>AGTSetDataField</td>
<td></td>
</tr>
<tr>
<td>AGTSetNotifyKeyField</td>
<td></td>
</tr>
<tr>
<td>AGTUpdateField</td>
<td></td>
</tr>
<tr>
<td>Error text</td>
<td>E28892: No inbound calling list fields are available.</td>
</tr>
</tbody>
</table>

**E28893**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTClearDataSet</td>
<td></td>
</tr>
<tr>
<td>AGTListDataFields</td>
<td>No outbound calling list fields available. Retry with &lt;ListType&gt; I.</td>
</tr>
<tr>
<td>AGTReadField</td>
<td></td>
</tr>
<tr>
<td>AGTSetDataField</td>
<td></td>
</tr>
<tr>
<td>AGTSetNotifyKeyField</td>
<td></td>
</tr>
<tr>
<td>AGTUpdateField</td>
<td></td>
</tr>
<tr>
<td>Error text</td>
<td>E28893: No outbound calling list fields are available.</td>
</tr>
</tbody>
</table>
System messages

E28894

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetNotifyKeyField</td>
<td>&lt;FieldName&gt; not found. The &lt;FieldName&gt; 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.</td>
</tr>
<tr>
<td>AGTSetDataField</td>
<td>&lt;FieldName&gt; not found. The &lt;FieldName&gt; 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.</td>
</tr>
<tr>
<td>AGTRedField</td>
<td>&lt;FieldName&gt; not found. Retry with valid field name for the calling list.</td>
</tr>
<tr>
<td>AGTUpdateField</td>
<td>&lt;FieldName&gt; not found. The system did not find the field name specified in the calling list. Field names are case sensitive and may contain underscores (_) in place of spaces.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28894: &lt;FieldName&gt; not found.</td>
</tr>
</tbody>
</table>

E28895

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>The agent is already available for work.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28895: Already available for work.</td>
</tr>
</tbody>
</table>

E28896

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>The agent's headset must be active. Execute AGTConnHeadset and retry.</td>
</tr>
<tr>
<td>AGTReadyNextItem</td>
<td>Headset must be active. The agent's headset is disconnected. Reconnect the headset and retry.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28896: Headset must be active.</td>
</tr>
</tbody>
</table>
## E28897

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>An available for work request is already pending.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28897: Available for work request is pending.</td>
</tr>
</tbody>
</table>

## E28898

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>The job is not available for login. The job may have become</td>
</tr>
<tr>
<td></td>
<td>inactive since it was attached. Execute AGTListJobs to check</td>
</tr>
<tr>
<td></td>
<td>the job status.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28898: Job is not available.</td>
</tr>
</tbody>
</table>

## E28899

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>There is no available for work request pending. AGTAvailWork</td>
</tr>
<tr>
<td></td>
<td>did not execute. This message may indicate an Avaya Proactive</td>
</tr>
<tr>
<td></td>
<td>Contact system problem.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28899: No available for work request is pending.</td>
</tr>
</tbody>
</table>
### E28900 § <message>‡

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>An Avaya Proactive Contact system internal error occurred. Error message text follows.</td>
</tr>
<tr>
<td>AGTCallNotify</td>
<td></td>
</tr>
<tr>
<td>AGTDisconnHeadset</td>
<td></td>
</tr>
<tr>
<td>AGTHHeadsetConnBroken</td>
<td></td>
</tr>
<tr>
<td>AGTJobTransLink</td>
<td></td>
</tr>
<tr>
<td>AGTJobTransRequest</td>
<td></td>
</tr>
<tr>
<td>AGTPreviewRecord</td>
<td></td>
</tr>
<tr>
<td>AGTReceiveMessage</td>
<td></td>
</tr>
<tr>
<td>AGTManagedCall</td>
<td></td>
</tr>
<tr>
<td>Error text</td>
<td>E28900: Wrong message ID received - %s.</td>
</tr>
</tbody>
</table>

### E28901

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTJobTransRequest</td>
<td>Agent is not available for work. The AGTNoFurtherWork command initiated by agent is not necessary.</td>
</tr>
<tr>
<td>AGTManagedCall</td>
<td>Agent is not available for work. Retry after executing AGTAvailWork.</td>
</tr>
<tr>
<td>AGTReadyNextItem</td>
<td></td>
</tr>
<tr>
<td>Error text</td>
<td>E28901: Not available for work.</td>
</tr>
</tbody>
</table>

### E28902

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReadyNextItem</td>
<td>Already on a customer record. Release the current record with AGTReleaseLine or AGTFinishedItem and retry.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28902: Already have open customer record.</td>
</tr>
</tbody>
</table>
### E28903

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReadyNextItem</td>
<td>Already set ready for next customer record. AGTReadyNextItem already executed.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28903: Already set ready for next customer record.</td>
</tr>
</tbody>
</table>

### E28904

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReadyNextItem</td>
<td>Request for no further work pending. Command is disabled.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28904: Request for no further work is pending.</td>
</tr>
</tbody>
</table>

### E28905

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReadyNextItem</td>
<td>Request to transfer to another job is active. Retry after job transfer is complete.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28905: Request to transfer to another job is active.</td>
</tr>
</tbody>
</table>

### E28906

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTCallNotify</td>
<td>The agent is not ready for next customer record. Call notification event is inappropriate.</td>
</tr>
<tr>
<td>AGTPreviewRecord</td>
<td>The agent is not ready for next customer record. Preview record event is inappropriate.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28906: Not ready for next customer record.</td>
</tr>
</tbody>
</table>
System messages

### E28907

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTManagedCall</td>
<td>Attached job is not a Managed Dialing job. Command is only available for Managed Dialing jobs.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28907: Not a Managed Dialing job.</td>
</tr>
</tbody>
</table>

### E28908

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTManagedCall</td>
<td>Agent is not previewing a customer record. Retry after receiving a record to preview.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28908: No open customer record.</td>
</tr>
</tbody>
</table>

### E28909

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTManagedCall</td>
<td>Managed call is already complete. The AGTManagedCall command is already being executed. May occur when agent executes the command just as the time-out elapses.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28909: Managed dialing call is already complete.</td>
</tr>
</tbody>
</table>

### E28910

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTManagedCall</td>
<td>Managed call already canceled or complete. Either an AGTFinishedItem, AGTReleaseLine, or AGTManagedCall command is already executing. May 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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28910: Managed dialing call is cancelled or complete.</td>
</tr>
</tbody>
</table>
### E28911

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTManagedCall</td>
<td>Managed call already canceled.</td>
</tr>
<tr>
<td></td>
<td>AGTFinishedItem is already executing.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28911: Managed dialing call is canceled.</td>
</tr>
</tbody>
</table>

### E28912

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReadField</td>
<td>Customer record is not available. The agent is not working</td>
</tr>
<tr>
<td></td>
<td>with a customer record.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28912: Customer record is not available for update.</td>
</tr>
</tbody>
</table>

### E28913

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTDetachJob</td>
<td>No job is attached. There is nothing to detach.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28913: There is no attached job to detach.</td>
</tr>
</tbody>
</table>

### E28914

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTDetachJob</td>
<td>Agent is still available for work on the job and calls are still being routed to the agent headset. Execute AGTNoFurtherWork and retry.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28914: Still available for work on the job.</td>
</tr>
</tbody>
</table>
## System messages

### E28915

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTDetachJob</td>
<td>Not logged out of the job yet. The AGTNoFurtherWork command is executing. Wait for a completion message from Avaya Proactive Contact and retry.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28915: Not logged out of job.</td>
</tr>
</tbody>
</table>

### E28916

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogoff</td>
<td>There is a job attached. Detach the job and retry.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28916: Job attached. Detach job and retry.</td>
</tr>
</tbody>
</table>

### E28917

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTJobTransLink</td>
<td>No job is attached. There is no current job that could have a job linked to it.</td>
</tr>
<tr>
<td>AGTJobTransRequest</td>
<td>No job is attached. The transfer request is inappropriate.</td>
</tr>
<tr>
<td>AGTNoFurtherWork</td>
<td>No job is attached; nothing to log out of.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28917: No job attached. Job linking is not available.</td>
</tr>
</tbody>
</table>

### E28918

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTNoFurtherWork</td>
<td>Not available for work on the job. Agent is not logged in to a job; nothing to log out of.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28918: Not available for work on the job.</td>
</tr>
</tbody>
</table>
### E28919

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTFinishedItem</td>
<td>The agent is not currently working with a customer record. There is no customer record to release.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28919: No active customer record to release.</td>
</tr>
</tbody>
</table>

### E28920 § <HeadsetID>‡

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTConnHeadset</td>
<td>&lt;HeadsetID&gt; is not in the reserved list. Retry the command with a valid reserved headset ID.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28920: Headset ID is not found in reserved list - %s.</td>
</tr>
</tbody>
</table>

### E28921

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSystemError</td>
<td>Fatal error - terminating. Avaya Proactive Contact has experienced a critical internal error and is shutting down.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28921: FATAL ERROR. AGENT PROCESS (agent binary) ENDING.</td>
</tr>
</tbody>
</table>

### E28922

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReserveHeadset</td>
<td>No reserve headset ID request pending. May indicate an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28922: No reserve headset ID request pending. Retry command with headset ID.</td>
</tr>
</tbody>
</table>
### E28923 § <HeadsetID>‡

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReserveHeadset</td>
<td>The requested headset ID is reserved.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28923: Headset ID ‡ is already reserved.</td>
</tr>
</tbody>
</table>

### E28924

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSystemError</td>
<td>Must sign in system first. Must execute AGTLogon before any other command.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28924: Must log into system first.</td>
</tr>
</tbody>
</table>

### E28925

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogon</td>
<td>This agent application is already logged in to the system from this session. To change the &lt;AgentName&gt; execute AGTLogoff and retry.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28925: Already logged on to the system.</td>
</tr>
</tbody>
</table>

### E28926

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogon</td>
<td>Invalid sign-on. Indicates that the login name or password is invalid.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28926: Invalid logon.</td>
</tr>
</tbody>
</table>
### **E28942**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTMoFlashBlind</td>
<td>Transfer job is not available. The <code>&lt;JobName&gt;</code> specified is not running, does not exist, or is not an inbound or blend job.</td>
</tr>
<tr>
<td>AGTMoFlashSupv</td>
<td></td>
</tr>
<tr>
<td>Error text</td>
<td>E28942: Transfer job is not available.</td>
</tr>
</tbody>
</table>

### **E28946**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>Agent not yet acquired. The agent is logged in as an Agent Blending agent, but Avaya Proactive Contact has not acquired the agent for outbound calling.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28946: Predictive Blend agent is not acquired for outbound calls.</td>
</tr>
</tbody>
</table>

### **E28947**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTFinishedItem</td>
<td>Completion code is invalid. Retry command with a code returned by AGTListKeys for the current job.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28947: Invalid completion code.</td>
</tr>
</tbody>
</table>

### **E28950**

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogonAcid</td>
<td>The specified extension is not one of the ACD extensions defined on Avaya Proactive Contact.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28950: Extension not a valid ACD extension.</td>
</tr>
</tbody>
</table>
## System messages

### E28951

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogonAcd</td>
<td>The specified extension is in use by another agent.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28951: Extension is in use by another agent.</td>
</tr>
</tbody>
</table>

### E28952

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogonAcd</td>
<td>Cannot add another agent; the maximum number of ACD agents are already logged in.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28952: Cannot add agent. Maximum number of ACD agents logged on.</td>
</tr>
</tbody>
</table>

### E28953

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogonAcd</td>
<td>Could not read the file containing the ACD extensions. Indicates an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28953: Cannot read file containing ACD extensions.</td>
</tr>
</tbody>
</table>

### E28954

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogonAcd</td>
<td>Duplicate login - Please try again. Indicates that the agent login name is already logged in to dispatcher.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28954: Duplicate login. Please try again.</td>
</tr>
</tbody>
</table>
### E28955

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogonAcd</td>
<td>The Agent Blending dispatcher process is not running.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28955: Predictive Blend dispatcher process is not running on Avaya Proactive Contact.</td>
</tr>
</tbody>
</table>

### E28956

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogonAcd</td>
<td>ACD login error. Dispatcher returned an “unknown” response to the agent application.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28956: Unknown ACD logon error.</td>
</tr>
</tbody>
</table>

### E28964

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReadyNextItem</td>
<td>The agent phone is busy.</td>
</tr>
<tr>
<td>AGTManagedCall</td>
<td></td>
</tr>
<tr>
<td>Error text</td>
<td>E28964: Agent phone is busy. Release phone line before proceeding.</td>
</tr>
</tbody>
</table>

### E28965

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReadyNextItem</td>
<td>CTI link is down.</td>
</tr>
<tr>
<td>AGTManagedCall</td>
<td></td>
</tr>
<tr>
<td>Error text</td>
<td>E28965: Softdialer link is down. Please wait until the link is up.</td>
</tr>
</tbody>
</table>
### System messages

#### E28967

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTNoFurtherWork</td>
<td>An agent attempted to log out of an agent-owned recall without logging out of the job from which the agent transferred.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28967: An agent is not allowed to logoff.</td>
</tr>
</tbody>
</table>

#### E28812

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogon</td>
<td>&lt;AgentName&gt; is logged in to another application process. The agent may be logged in at another location. Retry login with a different &lt;AgentName&gt;.</td>
</tr>
<tr>
<td>Error text</td>
<td>E28812: Agent %s already logged on. Access is denied.</td>
</tr>
</tbody>
</table>

#### E29000

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>Only Managed agents may join this job. The agent application agent type (work class) does not match the job type. Reset the agent type to M and retry.</td>
</tr>
<tr>
<td>Error text</td>
<td>E29000: Agent type is not M. Cannot join Managed Dialing job.</td>
</tr>
</tbody>
</table>

#### E29203

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSystemError</td>
<td>Could not attach slot.</td>
</tr>
</tbody>
</table>
### E29206 § <unit>‡

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAttachJob</td>
<td>Cannot attach the <code>&lt;unit&gt;</code> segment of shared memory that Avaya Proactive Contact is using for the job. Most jobs use the same unit of shared memory, defined in a configuration file on the Avaya Proactive Contact system. This message indicates a problem with the Avaya Proactive Contact configuration.</td>
</tr>
<tr>
<td>Error text</td>
<td>E29206: System error. Failed to attach %s segment of shared memory.</td>
</tr>
</tbody>
</table>

### E29950

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTHoldCall</td>
<td>This command is not available on an Avaya Proactive with CTI system.</td>
</tr>
<tr>
<td>AGTUnholdCall</td>
<td></td>
</tr>
<tr>
<td>AGTManualCall</td>
<td></td>
</tr>
<tr>
<td>AGTHookFlashLine</td>
<td></td>
</tr>
<tr>
<td>AGTTransferCall</td>
<td></td>
</tr>
<tr>
<td>AGTDialDigit</td>
<td></td>
</tr>
<tr>
<td>AGTAdjustHeadset</td>
<td></td>
</tr>
<tr>
<td>Error text</td>
<td>E29950: Feature not available in Softdialer Mode</td>
</tr>
</tbody>
</table>

### E50100

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogon</td>
<td>The maximum number of agents allowed by Agent_count are already logged in to the system.</td>
</tr>
<tr>
<td>Error text</td>
<td>E50100: Exceeded the maximum number of agents allowed.</td>
</tr>
</tbody>
</table>
### E50611 § <HeadsetID>‡

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReserveHeadset</td>
<td>The requested headset ID is in use.</td>
</tr>
<tr>
<td>Error text</td>
<td>E50611: Headset ID is already reserved.</td>
</tr>
</tbody>
</table>

### E50612 § <HeadsetID>‡

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReserveHeadset</td>
<td>No more headsets are permitted on the system.</td>
</tr>
<tr>
<td>Error text</td>
<td>E50612: No more headsets permitted on the system.</td>
</tr>
</tbody>
</table>

### E50613 § <HeadsetID>‡

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTReserveHeadset</td>
<td>Failure to access the headset ID table. May indicate a</td>
</tr>
<tr>
<td></td>
<td>permissions problem for the headset ID table file on the</td>
</tr>
<tr>
<td></td>
<td>Avaya Proactive Contact system, a corrupt headset ID table</td>
</tr>
<tr>
<td></td>
<td>file, or an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>Error text</td>
<td>E50613: Failed to access the headset ID file.</td>
</tr>
</tbody>
</table>

### E70000

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic</td>
<td>Incorrect number of arguments. The command issued from the</td>
</tr>
<tr>
<td></td>
<td>agent application contained the wrong number of arguments.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70000: Incorrect number of arguments.</td>
</tr>
</tbody>
</table>
### E70001

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic</td>
<td>Incorrect message type (not Data, Pending, Error, Response, Busy, or Notify). This code may indicate that the agent application message headers are not formatting correctly.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70001: Incorrect message type.</td>
</tr>
</tbody>
</table>

### E70002, <message>

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic</td>
<td>Pending request timed out waiting for &lt;message&gt;. The &lt;message&gt; is a text string containing the command that timed out and the response that the agent was expecting from Avaya Proactive Contact. This code may indicate an Avaya Proactive Contact system problem.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70002: Timed out waiting for %s.</td>
</tr>
</tbody>
</table>

### E70003

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTListJobs</td>
<td>Unknown job type. Retry with a different job type.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70003: Unknown job type.</td>
</tr>
</tbody>
</table>

### E70006

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAvailWork</td>
<td>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.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70006: Need to select work unit.</td>
</tr>
</tbody>
</table>
### E70007

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTMoFlashBlind</td>
<td>Cannot transfer an inbound call. AGTMoFlashBlind can only transfer outbound calls.</td>
</tr>
<tr>
<td>AGTMoFlashSupv</td>
<td>Cannot transfer an inbound call. AGTMoFlashSupv can only transfer outbound calls.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70007: Cannot transfer an inbound call.</td>
</tr>
</tbody>
</table>

### E70008

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTMoFlashBlind</td>
<td>Must specify a transfer job. There is no default transfer job configured on Avaya Proactive Contact, so you must include a transfer &lt;JobName&gt; with a command.</td>
</tr>
<tr>
<td>AGTMoFlashSupv</td>
<td>Must specify a transfer job. There is no default transfer job configured on Avaya Proactive Contact, so you must include a transfer &lt;JobName&gt; with a command.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70008: Must specify a transfer job.</td>
</tr>
</tbody>
</table>

### E70009 <process name>

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTMoFlashBlind</td>
<td>Unable to send a message to &lt;process name&gt;. Internal Avaya Proactive Contact system error; agent cannot send messages.</td>
</tr>
<tr>
<td>AGTMoFlashSupv</td>
<td>Unable to send a message to &lt;process name&gt;. Internal Avaya Proactive Contact system error; agent cannot send messages.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70009: Unable to send the message.</td>
</tr>
</tbody>
</table>

### E70010

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTMoFlashSupv</td>
<td>Conference is in progress. The agent executed the command a third time after the conference call has begun.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70010: Conference call is already in progress.</td>
</tr>
</tbody>
</table>
### E70011

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTLogonAcd</td>
<td>Agent Blending is not available on this Avaya Proactive Contact system. This system cannot work with ACD agents.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70011: Predictive Blend is not available on this system.</td>
</tr>
</tbody>
</table>

### E70012

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetPassword</td>
<td>The agent’s current password is no longer valid. Enter a new password to continue the log on.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70012: Password has expired, must be changed.</td>
</tr>
</tbody>
</table>

### E70013

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetPassword</td>
<td>Avaya Proactive Contact is not configured to allow an agent to change his or her password.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70013: Password can not be changed, change limit not expired.</td>
</tr>
</tbody>
</table>

### E70014

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetPassword</td>
<td>Avaya Proactive Contact is configured with a locked password file. The agent cannot change his or her password.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70014: Password can not be changed, password file locked.</td>
</tr>
</tbody>
</table>
System messages

### E70015

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetPassword</td>
<td>The agent does not have access privileges to set a password*. Contact your system administrator to reset a password.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70015: Unable to become root privilege for setting password.</td>
</tr>
</tbody>
</table>

### E70016

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetPassword</td>
<td>The password the agent entered is not correct. Retry using a different password that follows the password* rules.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70016: Original password is invalid.</td>
</tr>
</tbody>
</table>

### E70017

<table>
<thead>
<tr>
<th>Source</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTSetPassword</td>
<td>The new password entry is incorrect. Retry using a different password that follows the password* rules.</td>
</tr>
<tr>
<td>Error text</td>
<td>E70017: New password entered is invalid.</td>
</tr>
</tbody>
</table>

*See [AGTSetPassword](#) on page 226 for password rules.*
# Data messages

<table>
<thead>
<tr>
<th>Message Number</th>
<th>Command(s)</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>M00000</td>
<td>Generic</td>
<td>Complete.</td>
</tr>
<tr>
<td></td>
<td>Error text</td>
<td>M00000: Successful completion.</td>
</tr>
<tr>
<td>M000001 §</td>
<td>AGTListKeys</td>
<td>Data message. Completion codes are 3-digit numbers. Customers define call completion codes during specification of their Avaya Proactive Contact system. The descriptions are from the Avaya Proactive Contact compcode.cfg file. Telephone script labels are found in the Avaya Proactive Contact telephny.spt file. The system associates the labels with the respective call completion codes.</td>
</tr>
<tr>
<td>&lt;CompCode&gt;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;Description&gt;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;ScriptLabel&gt; §</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;CompCode&gt;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;Description&gt;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;ScriptLabel&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>§…‡</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M00001 §</td>
<td>AGTListDataFields</td>
<td>Data message. The &lt;FieldType&gt; is C (character), N (numeric), $(currency), D (date) or T (time). The F value is a placeholder for future use.</td>
</tr>
<tr>
<td>&lt;FieldName&gt;,</td>
<td>AGTListCallFields</td>
<td></td>
</tr>
<tr>
<td>&lt;FieldLength&gt;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;FieldType&gt;, F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>§…‡</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M00001 §</td>
<td>AGTManagedCall</td>
<td>Data message. &lt;CallStatus&gt; should always be (CONNECT).</td>
</tr>
<tr>
<td>&lt;CallStatus&gt;‡</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M00001 §</td>
<td>AGTListJobs</td>
<td>Data message containing requested job information. Job status appears as a single character, I for inactive or A for active.</td>
</tr>
<tr>
<td>&lt;JobType&gt;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;JobName&gt;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;Status&gt; §</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;JobType&gt;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;JobName&gt;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;Status&gt;…‡</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M00001 §</td>
<td>AGTReceiveMessage</td>
<td>Data message. The text of the message the supervisor sent.</td>
</tr>
<tr>
<td>&lt;Message&gt;‡</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M00001 §</td>
<td>AGTListUnits</td>
<td>Data message. There are as many data elements as there are unit ID values on the job.</td>
</tr>
<tr>
<td>&lt;UnitID&gt;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;UnitID&gt;…‡</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 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>… | AGTNotify AGTPreviewRecord | Additional fields data message.

M00001§<Format>§<Phones> | AGTListCallbackFmt | Data message. `<Format>` is the date format used on Avaya 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> | AGTNotify | Initial notification data message.

M00001 | Error text | M00001: Data message.
Pending messages

In the information that follows, each message number has its own heading.

**S28814**

<table>
<thead>
<tr>
<th>Commands</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTAdjustHeadset</td>
<td>Transfer is in progress. The last call is being transferred. Retry the command after connecting to the next call.</td>
</tr>
<tr>
<td>AGTDialDigit</td>
<td></td>
</tr>
<tr>
<td>AGTHookflashLine</td>
<td>The transfer for the last call is in progress. Retry after the transfer is complete.</td>
</tr>
<tr>
<td>AGTGetHeadsetVol</td>
<td>A transfer is in progress. The customer call is still being transferred to the agent. Retry after connecting to the call.</td>
</tr>
<tr>
<td>AGTHoldCall</td>
<td>Transfer is in progress. Avaya Proactive Contact cannot place the call on hold.</td>
</tr>
<tr>
<td>AGTManualCall</td>
<td>Transfer in progress. The last call is being transferred. Retry after transfer complete.</td>
</tr>
<tr>
<td>AGTMoFlashBlind</td>
<td>Trunk-to-trunk transfer is already in progress. May mean the agent previously executed AGTTransferCall.</td>
</tr>
<tr>
<td>AGTMoFlashSupv</td>
<td></td>
</tr>
<tr>
<td>AGTUnholdCall</td>
<td>Transfer is in progress. The call is being transferred.</td>
</tr>
<tr>
<td>Error text</td>
<td>S28814: Transfer is in progress.</td>
</tr>
</tbody>
</table>

**S28833**

<table>
<thead>
<tr>
<th>Commands</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic</td>
<td>Pending.</td>
</tr>
<tr>
<td>Error text</td>
<td>S28833: Request is pending.</td>
</tr>
</tbody>
</table>
System messages

**S28971**

<table>
<thead>
<tr>
<th>Commands</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTlicbFeNotif</td>
<td>Standby to take outbound calls... Acquisition to outbound is pending.</td>
</tr>
<tr>
<td>Error text</td>
<td>S28971: Standby to take outbound calls...</td>
</tr>
</tbody>
</table>

**S28972**

<table>
<thead>
<tr>
<th>Commands</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTlicbFeNotif</td>
<td>Standby to take inbound calls... Release to inbound is pending.</td>
</tr>
<tr>
<td>Error text</td>
<td>S28972: Standby to take inbound calls...</td>
</tr>
</tbody>
</table>

**S28833**

<table>
<thead>
<tr>
<th>Commands</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic</td>
<td>Pending.</td>
</tr>
<tr>
<td>Error text</td>
<td>S28833: Request is pending.</td>
</tr>
</tbody>
</table>

**S70000, <JobName>**

<table>
<thead>
<tr>
<th>Commands</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTListState</td>
<td>Agent is on a call. Working on &lt;JobName&gt;, currently working with a customer record.</td>
</tr>
<tr>
<td>Error text</td>
<td>S70000: Working on job %s. An agent is using a customer record.</td>
</tr>
</tbody>
</table>
### S70001, `<JobName>`

<table>
<thead>
<tr>
<th>Commands</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTListState</td>
<td>Agent is ready for a call. Working on <code>&lt;JobName&gt;</code>, currently waiting for next customer record.</td>
</tr>
<tr>
<td>Error text</td>
<td>S70001: Working on job %s. Waiting for the next customer record.</td>
</tr>
</tbody>
</table>

### S70002, `<JobName>`

<table>
<thead>
<tr>
<th>Commands</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTListState</td>
<td>Agent has joined (logged in to) a job. Working on <code>&lt;JobName&gt;</code>, currently available for work but not ready for next item.</td>
</tr>
<tr>
<td>Error text</td>
<td>S70002: Working on job %s. Not ready for next customer record.</td>
</tr>
</tbody>
</table>

### S70003, `<JobName>`

<table>
<thead>
<tr>
<th>Commands</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTListState</td>
<td>Agent has selected a job to work with. Attached to <code>&lt;JobName&gt;</code>, but not yet available for work.</td>
</tr>
<tr>
<td>Error text</td>
<td>S70003: Job %s attached. Not yet ready for work.</td>
</tr>
</tbody>
</table>

### S70004

<table>
<thead>
<tr>
<th>Commands</th>
<th>Message Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGTListState</td>
<td>Agent logged in to Avaya Proactive Contact. Agent is idle, not yet attached to a job.</td>
</tr>
<tr>
<td>Error text</td>
<td>S70004: No job attached. Agent is idle.</td>
</tr>
</tbody>
</table>
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 Avaya Proactive Contact), see Using Avaya Proactive Contact Agent API DLL on page 61.

This section contains the following topics:

- IServerStartup interface on page 301
- IConfigure interface on page 302
- IMoagent interface on page 303

IServerStartup interface

This interface is a COM interface. Client applications access Moagent32.dll through the properties in this interface.

Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropIConfig() As IConfigure</td>
<td>Returns a pointer to the IConfigure interface.</td>
</tr>
<tr>
<td>PropNexus() As IMoagent</td>
<td>Returns a pointer to the IMoagent interface.</td>
</tr>
<tr>
<td>PropWinHwnd</td>
<td>Provides Moagent32.dll with a client window handle for post-message event notification. (Use for PowerBuilder client applications only.)</td>
</tr>
<tr>
<td>PropWinUserMsg</td>
<td>Provides Moagent32.dll with a client window handle for post-message event notification. (Use for PowerBuilder client applications only.)</td>
</tr>
</tbody>
</table>

Events

None.

Methods

None.
**IConfigure interface**

This interface is public usable.

**Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetStatFileName</td>
<td>When set to “True,” Moagent32.dll writes statistics to the file name you specify (can include path).</td>
</tr>
<tr>
<td>SetErrFileName</td>
<td>When set to “True,” Moagent32.dll creates an error file consisting of the last N errors (identified by SetNumOfLstErrs).</td>
</tr>
<tr>
<td>SetNumOfLstErrs</td>
<td>Specifies the number of errors to include in the errors display and error file (first in, first out).</td>
</tr>
<tr>
<td>SetCreateLogFile</td>
<td>When set to “True,” Moagent32.dll creates a message transaction log file (identified by SetLogFileName).</td>
</tr>
<tr>
<td>SetLogFileName</td>
<td>Specifies the name of the message transaction log file.</td>
</tr>
<tr>
<td>SetLogonRecovery</td>
<td>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.</td>
</tr>
<tr>
<td>SetReserveConnHeadSet</td>
<td>When set to “True,” Moagent32.dll automatically reserves and connects a headset after login if a headset value is given.</td>
</tr>
<tr>
<td>SetLogonTimeout</td>
<td>Specifies how long Moagent32.dll waits (in seconds) before sending a message advising that the Avaya Proactive Contact login attempt timed out. The timeout range is between 2 and 200 seconds.</td>
</tr>
</tbody>
</table>
Events

None.

Methods

None.

**IMOagent interface**

This interface is public usable. See Chapter 2: Using Avaya Proactive Contact Agent API DLL on page 61 for information on using the events and methods associated with the IMOagent interface in your client application. See Chapter 3: Commands and notification events on page 83 for descriptions of the Avaya Proactive Contact API commands associated with the methods listed below.

Properties

None.

Event

Avaya MosaixEvent (ErrFlag As Boolean, NotifyType As String,

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetAvayaMosaixTimeout</td>
<td>Specifies how long Moagent32.dll waits (in seconds) before sending a message advising that the Avaya Proactive Contact command failed. The timeout range is between 2 and 200 seconds.</td>
</tr>
<tr>
<td>SetUseDII Dbs</td>
<td>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:</td>
</tr>
</tbody>
</table>
|                                 | ● AttachJob
|                                 | ● FinishedItem
|                                 | ● SetWorkClass
|                                 | ● AdjustHeadset
|                                 | ● DialDigit
|                                 | ● SendMessage
|                                 | ● SetCallback
|                                 | ● SetPassword
|                                 | ● SetUnit
|                                 | ● TransferCall
Moagent32.dll interfaces

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)
- DisconnectHeadset (ErrCode,ErrText)
- DoNotCall (ErrCode,ErrText)
- EchoOff (ErrCode,ErrText)
- EchoOn (ErrCode,ErrText)
- FreeHeadset (ErrCode,ErrText)
- HangupCall (ErrCode,ErrText)
- HoldCall (ErrCode,ErrText)
- LogIoStart (ErrCode,ErrText)
- LogIoStop (ErrCode,ErrText)
- Logoff (ErrCode,ErrText)
- LogoffAcd (ErrCode,ErrText)
- NoFurtherWork (ErrCode,ErrText)
- ReadyNextItem (ErrCode,ErrText)
- UnholdCall (ErrCode,ErrText)

The following methods have no arguments that return data:

- ListCallbackFmt (ErrCode,ErrText)
- GetHeadsetVol (ErrCode,ErrText)
- ListCallLists (ErrCode,ErrText)
- ListKeys (ErrCode,ErrText)
- ListState (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)
● 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)

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).
Moagent32.dll interfaces
Appendix E: Data parameters

This appendix lists data parameters in alphabetical order.

Note:
The agent binary returns some of the same parameters to the agent application in data records or as part of the Avaya Proactive Contact system error messages.

<table>
<thead>
<tr>
<th>Data Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;AgentName&gt;</td>
<td>The user name of the calling agent. Use a maximum of 19 characters, 7-bit USASCII character set. Do not include embedded spaces.</td>
</tr>
<tr>
<td>&lt;CallType&gt;</td>
<td>Identifies the call source. If Avaya 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.</td>
</tr>
<tr>
<td>&lt;ClassID&gt;</td>
<td>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).</td>
</tr>
<tr>
<td>&lt;CompCode&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>&lt;Date&gt;</td>
<td>The date to place the recall. CCYY/MM/DD, MM/DD/CCYY, or DD/MM/CCYY. Must match the format returned by AGTListCallbackFmt.</td>
</tr>
<tr>
<td>&lt;Digit&gt;</td>
<td>The digit for Avaya Proactive Contact to call. Use one character: a single number (0 - 9), *, or #. If the Avaya Proactive Contact master.cfg file contains values for the DIAL_STAR and DIAL_POUND variables, Avaya Proactive Contact substitutes those values for the normal * or # tones.</td>
</tr>
<tr>
<td>&lt;EarMouth&gt;</td>
<td>The headset volume setting to change: E (ear) or M (mouth).</td>
</tr>
<tr>
<td>&lt;ErrorMessage&gt;</td>
<td>The Avaya Proactive Contact error message number and any data or information that is part of that message.</td>
</tr>
<tr>
<td>&lt;Extension&gt;</td>
<td>The ACD extension to associate with the agent’s telephone. Use a numeric value, up to 20 characters.</td>
</tr>
<tr>
<td>&lt;FieldName&gt;</td>
<td>A field name from the calling list, as shown by AGTListDataFields. Up to 19 characters, alphanumeric, case sensitive, no embedded spaces.</td>
</tr>
<tr>
<td>&lt;FieldData&gt;</td>
<td>Value of data field from the customer record.</td>
</tr>
</tbody>
</table>
### Data Parameter Description

<table>
<thead>
<tr>
<th>Data Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;FileName&gt;</td>
<td>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 &lt;FileName&gt; parameter is case sensitive.</td>
</tr>
<tr>
<td>&lt;HeadsetID&gt;</td>
<td>Identifies the extension for the agent’s headset. Up to 14 numeric characters, no special characters or embedded spaces.</td>
</tr>
<tr>
<td>&lt;JobName&gt;</td>
<td>The name of a job on Avaya Proactive Contact. Use up to 19 characters, 7-bit USASCII. The &lt;JobName&gt; is case sensitive and may not include special characters or embedded spaces.</td>
</tr>
<tr>
<td>&lt;JobType&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>&lt;ListName&gt;</td>
<td>The name of the calling list. Use alphanumeric values, 7-bit USASCII. Calling list names are case sensitive.</td>
</tr>
<tr>
<td>&lt;ListType&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>&lt;Message&gt;</td>
<td>The text of the message from the supervisor or to display on the supervisor’s screen. The message may be up to 79 characters long. Embedded spaces and special characters are allowed.</td>
</tr>
<tr>
<td>&lt;MessageNo&gt;</td>
<td>Value is the message number to play to the customer. These numbers must appear in the voicemsg.cfg file on the Avaya Proactive Contact system. Use up to three digits, ranging from 1 to 255.</td>
</tr>
<tr>
<td>&lt;NewValue&gt;</td>
<td>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).</td>
</tr>
<tr>
<td>&lt;NotifyFieldData&gt;</td>
<td>The notification key field value in the customer’s record.</td>
</tr>
<tr>
<td>&lt;NotifyFieldName&gt;</td>
<td>The calling list field name requested by AGTSetNotifyKeyField.</td>
</tr>
<tr>
<td>&lt;OpMesg&gt;</td>
<td>Messages from call or preview notification containing field information from the customer record. The field choice is configured in alljobs.dat on the Avaya Proactive Contact system. The default field is NAME. If the call is a voice and data transfer from another agent, the &lt;OpMesg&gt; is “TRANSFER CALL.” If the notification is a Managed Dialing preview record, the text string “Preview” always follows the information.</td>
</tr>
<tr>
<td>&lt;Password&gt;</td>
<td>Enter the agent’s password. Use two to eight characters, 7-bit USASCII character set. Do not include embedded spaces.</td>
</tr>
<tr>
<td>Data Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>&lt;PBX ID&gt;</td>
<td>The identification of the PBX (switch) used for Agent Blending. Default value is 1. Value given here must match the DBKGROUP parameter in the Avaya Proactive Contact master.cfg file.</td>
</tr>
<tr>
<td>&lt;PhoneIndex&gt;</td>
<td>Index of the Avaya 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.</td>
</tr>
<tr>
<td>&lt;PhoneNumber&gt;</td>
<td>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 Avaya Proactive Contact system.</td>
</tr>
<tr>
<td>&lt;RecallName&gt;</td>
<td>This is the customer name field to contact during the recall. There may be a default calling list field name to use for recalls specified in the job’s *.fdict file. This parameter is optional.</td>
</tr>
<tr>
<td>&lt;RecallNumber&gt;</td>
<td>This is the telephone number to use for the recall (instead of the Phone Index specification that can be entered or left blank.) There may 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 Avaya Proactive Contact system.</td>
</tr>
<tr>
<td>&lt;ScriptLabel&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>&lt;Time&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>&lt;UnitID&gt;</td>
<td>The unit ID value selected by the agent.</td>
</tr>
<tr>
<td>&lt;Volume&gt;</td>
<td>The numeric value for the headset volume setting. Whole numbers only, from 1 to 8.</td>
</tr>
<tr>
<td>&lt;WaitMsg&gt;</td>
<td>If present, an asterisk (*) separates &lt;WaitMsg&gt; from &lt;OpMesg&gt;. &lt;WaitMsg&gt; is a text string defined in the Avaya Proactive Contact job file indicating how long the customer has been on hold. &lt;WaitMsg&gt; may 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”).</td>
</tr>
</tbody>
</table>
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
```

**Note:**
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.)
Use the Agent DDE Interface

**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.

**Note:**
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:

1. by pressing the Save button
2. by switching to a different screen
3. 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:

<table>
<thead>
<tr>
<th>Command line flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-r read-only</td>
<td>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.</td>
</tr>
<tr>
<td>-c [filename]</td>
<td>Explicitly points the agent to an ini file in a different directory. The default is a file named PCAgent.ini in the PCAgent directory.</td>
</tr>
<tr>
<td>Command line flag</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>-l [filename]</td>
<td>Explicitly specifies the path and filename of the log file.</td>
</tr>
<tr>
<td>-sl [filename]</td>
<td>Explicitly specifies the path and filename of the socket log file.</td>
</tr>
<tr>
<td>-nc</td>
<td>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.</td>
</tr>
<tr>
<td>-d [IP address]</td>
<td>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.</td>
</tr>
</tbody>
</table>
Use the Agent DDE Interface
Index

A

acquired phone numbers ........................................ 20
agent
call completion codes ......................................... 247
Agent API messages ............................................. 27
Agent application ................................................. 24
Agent application initiated messages ......................... 28
agent application session sample .............................. 41
Agent applications ............................................... 41
Agent binary .......................................................... 24
agent owned recall sample ..................................... 55
Agent states .......................................................... 32
Agent types .......................................................... 16
Agent_state .......................................................... 81
AgentName ........................................................... 307
AGTAdjustHeadset ................................................. 84, 86, 128, 163
AGTAvailWork ........................................................ 91
AGTCallNotify ........................................................ 94
AGTClearDataSet .................................................... 98
AGTConnHeadset .................................................... 100
AGTDetachJob ........................................................ 102
AGTDialDigit .......................................................... 104
AGTDisconnHeadset ............................................... 108
AGTDoNotCall ....................................................... 110
AGTDumpData ........................................................ 111
AGTEchoOff ........................................................... 116
AGTEchoOn .............................................................. 118
AGTFinishedItem ..................................................... 119
AGTFreeHeadset ...................................................... 121
AGTGetHeadsetVol ................................................... 123
AGTHangupCall ....................................................... 127
AGTHangupAdd ...................................................... 129
AGTHoldCall ........................................................... 131
AGTHookflashLine .................................................... 133
AGTIlcAbort .......................................................... 135
AGTIlcFeNotif ........................................................ 137
AGTIlcOffline ........................................................ 139
AGTIlcOnline ........................................................ 141
AGTJobEnd .............................................................. 143
AGTJobTransLink ..................................................... 145
AGTJobTransRequest ............................................... 147
AGTListCallbackFmt .............................................. 149
AGTListCallFields ................................................... 151
AGTListCallLists .................................................... 154
AGTListDataFields .................................................. 156
AGTListJobs ........................................................... 159
AGTListKeys .......................................................... 161
AGTListState .......................................................... 164
AGTListUnits .......................................................... 166
AGTLogloStart ...................................................... 168
AGTLogloStop ........................................................ 174
AGTLogoff .............................................................. 176
AGTLogoffAcD ......................................................... 178
AGTLogon .............................................................. 180
AGTLogonAcD ........................................................ 182
AGTMangedCall ........................................................ 185
AGTManualCall ........................................................ 187
AGTMoFlashBlind ..................................................... 189
AGTMoFlashSupv ..................................................... 193
AGTNoFurtherWork .................................................. 197
AGTPreviewRecord ................................................. 199
AGTReadField ........................................................ 202
AGTReadyNextItem ................................................... 205
AGTReceiveMessage ................................................ 207
AGTReleaseLine ...................................................... 209
AGTReserveHeadset ................................................. 212
AGTSendMessage ..................................................... 214
AGTSetCallback ....................................................... 216
AGTSetDataField ..................................................... 221
AGTSetNotifyKeyField .............................................. 224
AGTSetPassword ..................................................... 226
AGTSetUnit ............................................................. 228
AGTSetWorkClass ..................................................... 231
AGTSTART .............................................................. 234
AGTSystemError ...................................................... 236
AGTTransferCall ...................................................... 238
AGTUHoldCall ........................................................ 240
AGTUUpdateField ..................................................... 242
AGTXferCustHangup ................................................ 244
AGTXferTrunkHangup ............................................... 245
alternate phone numbers ..................................... 20
API ........................................................................ 63
Automate call handling ......................................... 26
Avaya Proactive Contact
  call completion codes ........................................ 247
Avaya Proactive Contact server ................................. 24
Avaya Proactive Contact system error messages ............ 257

B

Basic ................................................................. 65
Before beginning .................................................... 7
binary .................................................................. 24
binary initiated messages ...................................... 28
Blend calling .......................................................... 19

April 2006 315
message formats ................................................. 251
data .......................................................... 253
notification event ............................................. 255
response ........................................................ 253
Message headers .............................................. 29
message scenarios ............................................ 36
Message terminators ......................................... 31
MessageNo ..................................................... 308
messages                                             
  Avaya Proactive Contact system error .................... 257
Method syntax ................................................. 79
Methods ........................................................ 304
methods ......................................................... 67
Moagent32 interface .......................................... 78
Moagent32.dll .................................................. 64, 69
Moagent32.dll dialog boxes .................................. 75
Moagent32.ini settings ........................................ 80
Moagent32.log ................................................... 40

N
NewValue ....................................................... 308
Notification event message formats ......................... 255
Notification event messages .................................. 28
notification events ............................................. 69
NotifyFieldData ................................................ 308
NotifyFieldName ............................................... 308

O
OpMesg .......................................................... 308
Outbound calling ............................................. 18
Outbound job commands (by State) .......................... 35
Outbound job message scenarios ............................. 36

P
parameters                                             
data ............................................................... 307
Password ......................................................... 308
PBX ID .......................................................... 309
PhoneIndex ...................................................... 309
PhoneNumber .................................................... 309
Placing manual calls ......................................... 20
Power Builder .................................................... 65
processing calling lists ..................................... 11
properties ....................................................... 72
PropIConfig ....................................................... 301

R
recall ............................................................. 55
RecallName ...................................................... 309
RecallNumber .................................................... 309

recalls .......................................................... 20
records .......................................................... 21
requirements .................................................... 62
Response message formats ..................................... 253
Response messages ............................................. 27
result codes. See completion codes

S
Sample agent application session ........................... 41
Sample agent owned ........................................... 55
Sample managed dialing job .................................. 47
Sample unit work list job ..................................... 52
samples .......................................................... 72
Scheduling recalls ............................................. 20
ScriptLabel ....................................................... 309
separators ....................................................... 31
server ............................................................ 24
Server_return_codes .......................................... 81
SetAvayaPDSTimeout .......................................... 303
SetCreateLogFile .............................................. 302
SetCreateStatFile ............................................. 302
SetErrFileName ................................................ 302
SetLogFileName ................................................ 302
SetLogonRecovery ............................................. 302
SetLogonTimeout .............................................. 302
SetNumOfLstErrs .............................................. 302
SetReserveConnHeadSet ...................................... 302
SetCreateStatFile ............................................. 302
settings                                             
  moagent32 .................................................... 80
SetUseDllDbs ................................................... 303
Start the agent binary ....................................... 27
State ............................................................. 35
states ........................................................... 32
syntax ........................................................... 79
system
  call completion codes ...................................... 247

T
telnet ............................................................ 9
terminators ..................................................... 31
Terminology .................................................... 24
Testing applications ......................................... 26
Time ............................................................. 309
Transferring calls ............................................ 20
types ............................................................ 16

U
unit work list ................................................... 52
UnitID ............................................................ 309
## Index

### V

<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB</td>
<td>65</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>65</td>
</tr>
<tr>
<td>Visual C++</td>
<td>65</td>
</tr>
<tr>
<td>Volume</td>
<td>309</td>
</tr>
</tbody>
</table>

### W

<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>WaitMsg</td>
<td>309</td>
</tr>
<tr>
<td>Winsockerrors</td>
<td>81</td>
</tr>
<tr>
<td>Work_class</td>
<td>80</td>
</tr>
<tr>
<td>Wrap up calls</td>
<td>21</td>
</tr>
</tbody>
</table>