



TELEVANTAGE

DEVELOPER'S GUIDE

TELEVANTAGE 5.0

COPYRIGHT

© 2002 Artisoft, Inc. All rights reserved. This manual and the software described in it are copyrighted with all rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system or translated into any language in any form by any means without the written permission of Artisoft, Inc.

Portions © 1999, Microsoft Corporation. All rights reserved.

TRADEMARKS

Artisoft and TeleVantage are registered trademarks of Artisoft, Inc. Microsoft, Windows, and Windows NT are registered trademarks of Microsoft Corporation. SQL Server is a trademark of Microsoft Corporation. Adobe and Acrobat are registered trademarks of Adobe Systems Incorporated. Dialogic is a registered trademark of Intel Corporation. Other brand names, company names and product names are trademarks or registered trademarks of their respective companies.

LIMITED WARRANTY ON SOFTWARE

Artisoft warrants that (a) the Software will perform substantially in accordance with the accompanying written materials for a period of (30) days from the date of receipt. Any implied warranties on the Software is limited to thirty (30) days. Some states/jurisdictions do not allow limitations on duration of an implied warranty, so the above information may not apply to you.

CUSTOMER REMEDIES: Artisoft's and its suppliers' entire liability and your exclusive remedy shall be, at Artisoft's option, either (a) return of the price paid, or (b) repair or replacement of the Software that does not meet Artisoft's Limited Warranty and which is returned to Artisoft with a copy of your receipt. This Limited Warranty is void if failure of the Software has resulted from accident, abuse, or misapplication. Any replacement Software will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer.

ARTISOFT AND ITS SUPPLIERS DISCLAIM ALL OTHER WARRANTIES, REPRESENTATIONS, PROMISES AND GUARANTEES, EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, ON THE SOFTWARE, MEDIA, DOCUMENTATION OR RELATED TECHNICAL SUPPORT INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, PERFORMANCE AND FITNESS FOR A PARTICULAR PURPOSE. Artisoft WILL NOT BE LIABLE FOR ANY BUG, ERROR, OMISSION, DEFECT, DEFICIENCY OR NONCONFORMITY IN ANY SOFTWARE. AS A RESULT, THE SOFTWARE AND DOCUMENTATION IS LICENSED "AS IS", AND THE PURCHASER ASSUMES THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE.

IN NO EVENT WILL ARTISOFT OR ITS SUPPLIERS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, CONTINGENT, CONSEQUENTIAL OR SIMILAR DAMAGES OF ANY KIND RESULTING FROM ANY DEFECT IN THE SOFTWARE OR DOCUMENTATION, INCLUDING, WITHOUT LIMITATION, ANY LOST PROFITS OR SAVINGS, DAMAGES FROM BUSINESS INTERRUPTION, LOSS OF OR TO DATA, COMPUTER PROGRAMS, BUSINESS, DOWNTIME, GOODWILL, DAMAGE TO OR REPLACEMENT OF EQUIPMENT OR PROPERTY, OR ANY COSTS OF RECOVERING, REPROGRAMMING OR REPRODUCING ANY PROGRAM OR DATA USED IN CONJUNCTION WITH THE PRODUCTS, EVEN IF Artisoft, ITS SUPPLIERS OR ANYONE ELSE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. YOU AGREE THAT Artisoft'S AND ITS SUPPLIERS' LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE AMOUNT PAID BY YOU FOR THIS PRODUCT. ANY WRITTEN OR ORAL INFORMATION OR ADVICE GIVEN BY Artisoft DEALERS, DISTRIBUTORS, AGENTS OR EMPLOYEES WILL IN NO WAY INCREASE THE SCOPE OF THIS WARRANTY, NOR MAY YOU RELY ON ANY SUCH WRITTEN OR ORAL COMMUNICATION. Some jurisdictions do not allow the limitation or exclusion of implied warranties or liability for incidental or consequential damages, and some jurisdictions have special statutory consumer protection provisions which may supersede this limitation, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from jurisdiction to jurisdiction.

Artisoft, Inc.
5 Cambridge Center
Cambridge, MA 02142

TABLE OF CONTENTS

Chapter 1. Introduction

| | |
|---|-----|
| About the TeleVantage SDK..... | 1-2 |
| Technical Support for the TeleVantage SDK..... | 1-2 |
| Development tools and terms..... | 1-3 |
| Installing the APIs..... | 1-3 |
| Installing the Device Status API and IVR Plug-in API..... | 1-3 |
| Installing the Client API..... | 1-3 |

Chapter 2. The Client API

| | |
|--|------|
| About the TeleVantage Client API..... | 2-2 |
| Client API object structures..... | 2-2 |
| Connecting to the Client API..... | 2-3 |
| Structure of a session..... | 2-4 |
| The System object..... | 2-5 |
| The User object..... | 2-6 |
| How folders are used..... | 2-7 |
| The Folder object..... | 2-8 |
| Default Folder structure..... | 2-9 |
| Other object structures..... | 2-10 |
| Agent objects..... | 2-11 |
| Call objects..... | 2-12 |
| CallHistory objects..... | 2-12 |
| CallRule objects..... | 2-13 |
| Contact objects..... | 2-13 |
| Dialing Service objects..... | 2-14 |
| Greeting objects..... | 2-15 |
| Message objects..... | 2-15 |
| Notification objects..... | 2-16 |
| PersonalStatus objects..... | 2-16 |
| RoutingList objects..... | 2-17 |
| Schedule objects..... | 2-18 |
| Shortcut objects..... | 2-18 |
| Station objects..... | 2-19 |
| StationFeature objects..... | 2-19 |
| Workgroup objects..... | 2-20 |
| Starting a new Client API project in Visual Basic..... | 2-20 |
| Programming tips and examples..... | 2-21 |

Chapter 3. The IVR Plug-in API

| | |
|--|-----|
| About the TeleVantage IVR Plug-in API..... | 3-2 |
| IVR Plug-in API Component and Sample Applications..... | 3-2 |
| How the IVR Plug-in components work..... | 3-3 |
| Monitoring standard and custom call data..... | 3-3 |
| IVR Plug-in licensing..... | 3-3 |
| Registering and Configuring IVR Plug-ins..... | 3-4 |
| Using the Customer ID IVR Plug-in..... | 3-6 |
| Managing the Customer ID database..... | 3-7 |
| Working with the Customer ID sample code..... | 3-7 |
| Running an IVR Plug-in that has a GUI..... | 3-8 |

| | |
|--|-----|
| Debugging IVR Plug-ins in Visual Basic | 3-8 |
| IVR Plug-in API quick reference | 3-9 |

Chapter 4. The Device Status API

| | |
|---|-----|
| About the TeleVantage Device Status API | 4-2 |
| How the Device Status components work | 4-2 |
| Using the Device Status components | 4-3 |
| Required runtime files | 4-4 |
| Device Status API quick reference | 4-5 |

Chapter 5. Using In-band Signaling

| | |
|--|-----|
| About in-band signaling with TeleVantage | 6-2 |
| TeleVantage telephone commands | 6-3 |
| Call handling menu | 6-3 |
| Quick call menu | 6-4 |
| Voice mail/Account menu commands | 6-5 |
| Quick commands for call center agents | 6-6 |

Chapter 6. The TeleVantage TAPI Service Provider

| | |
|--|-----|
| About the TeleVantage TAPI Service Provider | 5-2 |
| TeleVantage TAPI Service Provider capabilities | 5-2 |
| Supported TAPI functions | 5-2 |
| Supported bearer modes | 5-3 |
| Call states | 5-3 |
| Exported TSP functions | 5-3 |
| TAPI and TeleVantage custom call data | 5-3 |

Appendix A. IVR Plug-in API Reference

| | |
|--|------|
| Overview | A-2 |
| PluginCaller interface | A-2 |
| AssociateCalledExtension method (PluginCaller interface) | A-2 |
| AssociateCalledCallerID method (PluginCaller interface) | A-2 |
| AssociateCallerID method (PluginCaller interface) | A-3 |
| AssociateExtension method (PluginCaller interface) | A-4 |
| CallDone method (PluginCaller interface) | A-4 |
| Delete method (PluginCaller interface) | A-5 |
| DeviceName property (PluginCaller interface) | A-5 |
| Exists method (PluginCaller interface) | A-6 |
| Get method (PluginCaller interface) | A-6 |
| GetDevice method (PluginCaller interface) | A-8 |
| GetMedia method (PluginCaller interface) | A-9 |
| GetXmitTimeslot method (PluginCaller interface) | A-9 |
| Listen method (PluginCaller interface) | A-9 |
| Ping method (PluginCaller interface) | A-10 |
| PlugInComments property (PluginCaller interface) | A-10 |
| PlugInDID property (PluginCaller interface) | A-10 |
| PlugInExtension property (PluginCaller interface) | A-10 |
| PlugInName property (PluginCaller interface) | A-11 |
| PlugInVariable property (PluginCaller interface) | A-11 |
| ReleaseDevice method (PluginCaller interface) | A-11 |
| Set method (PluginCaller interface) | A-12 |
| SetStatus method (PluginCaller interface) | A-13 |
| PluginApplication interface | A-14 |
| CallOffering method (PluginApplication interface) | A-14 |
| CallPlaced method (PluginApplication interface) | A-14 |

| | |
|--|------|
| CallTerminated method (PluginApplication interface)..... | A-14 |
| IPluginServer interface | A-15 |
| PlaceCall method (IPluginServer interface) | A-15 |
| PluginMedia interface | A-16 |
| Dial method (PluginMedia interface)..... | A-16 |
| FlushDigitBuffer method (PluginMedia interface) | A-17 |
| GetCallProgress method (PluginMedia interface) | A-17 |
| GetDigits method (PluginMedia interface) | A-18 |
| GetXmitTimeslot method (PluginMedia interface) | A-19 |
| Listen method (PluginMedia interface) | A-19 |
| PlayFile method (PluginMedia interface) | A-20 |
| PlayString method (PluginMedia interface)..... | A-21 |
| PlayTone method (PluginMedia interface)..... | A-23 |
| RecordFile method (PluginMedia interface)..... | A-24 |
| SetRate method (PluginMedia interface) | A-25 |
| SetVolume method (PluginMedia interface) | A-25 |
| Stop method (PluginMedia interface)..... | A-25 |

Appendix B. Migrating Older IVR Plug-in Applications

| | |
|---|------|
| Overview..... | B-2 |
| IVRPlugInNotify2 interface..... | B-2 |
| CallOffering method (IVRPlugInNotify2 interface) | B-2 |
| CallPlaced method (IVRPlugInNotify2 interface) | B-3 |
| CallTerminated method (IVRPlugInNotify2 interface)..... | B-4 |
| IVRPlugIn2 object..... | B-4 |
| CallDone method (IVRPlugIn2 object) | B-5 |
| GetCustomPartyData method (IVRPlugIn2 object) | B-5 |
| GetPartyData method (IVRPlugIn2 object) | B-6 |
| GetXmitTimeslot method (IVRPlugIn2 object) | B-7 |
| Listen method (IVRPlugIn2 object) | B-7 |
| Ping method (IVRPlugIn2 object)..... | B-7 |
| PlaceCall method (IVRPlugIn2 object)..... | B-8 |
| Register method (IVRPlugIn2 object) | B-9 |
| SetCustomPartyData method (IVRPlugIn2 object)..... | B-10 |
| SetPartyData method (IVRPlugIn2 object) | B-10 |
| SetStatus method (IVRPlugIn2 object) | B-11 |
| Extension Property (IVRPlugIn2 object) | B-11 |
| DID Property (IVRPlugIn2 object) | B-11 |
| Name Property (IVRPlugIn2 object)..... | B-12 |
| Comments Property (IVRPlugIn2 object)..... | B-12 |
| CustomVariableName Property (IVRPlugIn2 object)..... | B-12 |
| CustomVariableValue Property (IVRPlugIn2 object) | B-12 |
| | B-12 |

Appendix C. Device Status API Reference

| | |
|--|-----|
| Overview..... | C-2 |
| Server object..... | C-2 |
| Connect method (Server object) | C-2 |
| Disconnect method (Server object)..... | C-2 |
| Devices Property (Server object) | C-2 |
| DeviceStateChanged Event (Server object) | C-3 |
| Device object | C-3 |
| Refresh method (Device object)..... | C-3 |
| AssignedExtensions Property (Device object)..... | C-3 |
| Class Property (Device object)..... | C-4 |

| | |
|---|-----|
| CurrentExtension Property (Device object)..... | C-4 |
| DefaultExtension Property (Device object) | C-4 |
| HookState Property (Device object)..... | C-5 |
| Name Property (Device object) | C-5 |
| Number Property (Device object)..... | C-5 |
| Status Property (Device object) | C-6 |
| Extension object | C-7 |
| Refresh method (Extension object)..... | C-7 |
| ACDDoNotDisturb Property (Extension object) | C-7 |
| DoNotDisturb Property (Extension object) | C-7 |
| Extension Property (Extension object)..... | C-7 |
| FirstName Property (Extension object) | C-8 |
| LastName Property (Extension object) | C-8 |

Appendix D. Client API Object Reference

| | |
|----------------------------------|------|
| Addresses object | D-3 |
| Agent object..... | D-4 |
| AgentStat object | D-5 |
| Agents object..... | D-5 |
| AudioClip object..... | D-6 |
| Call object..... | D-7 |
| CallHistory object..... | D-9 |
| CallRule object..... | D-10 |
| Column object..... | D-11 |
| Columns object..... | D-11 |
| Contact object..... | D-12 |
| Error object | D-13 |
| Field object | D-13 |
| Fields object..... | D-13 |
| Folder object..... | D-14 |
| Folders object | D-15 |
| Greeting object | D-15 |
| IAssociate object..... | D-16 |
| ICOMSecurity object..... | D-16 |
| IDialingService object | D-17 |
| IItem object | D-17 |
| IPhoneService object..... | D-18 |
| InternetService object..... | D-18 |
| LocaleCode object..... | D-19 |
| Items object | D-19 |
| Message object..... | D-20 |
| LocaleCodes object | D-20 |
| Notification object | D-21 |
| NotifyScheduleItem object..... | D-21 |
| NotifyScheduleItems object | D-22 |
| Parties object..... | D-22 |
| Party object..... | D-23 |
| PartyHistories object..... | D-24 |
| PartyHistory object..... | D-24 |
| Permissions object..... | D-25 |
| Permission object | D-25 |
| PersonalStatus object..... | D-26 |
| PhoneGatewayService object..... | D-27 |
| PhoneService object..... | D-28 |
| QueueByPeriodStat object | D-29 |

| | |
|-------------------------------------|------|
| QueueStat object | D-30 |
| QueueByShiftStat object..... | D-30 |
| Role object..... | D-32 |
| Roles object..... | D-32 |
| Recipients object | D-32 |
| RoutingList object | D-33 |
| RoutingListActions object | D-34 |
| RoutingListAction object | D-34 |
| RoutingService object..... | D-35 |
| RoutingListFinalAction object | D-35 |
| Schedule object | D-36 |
| ScheduledDate object..... | D-36 |
| ScheduledDates object..... | D-36 |
| ScheduledDay object..... | D-37 |
| ScheduledDays object | D-37 |
| Session object | D-38 |
| Schedules object | D-38 |
| Shortcut object..... | D-43 |
| ShortcutGroups object..... | D-44 |
| ShortcutGroup object..... | D-44 |
| Station object..... | D-45 |
| Shortcuts object..... | D-45 |
| StationButton object..... | D-46 |
| StationButtons object..... | D-46 |
| StationFeature object..... | D-47 |
| StationFeatures object..... | D-47 |
| StationParameters object | D-48 |
| StationType object..... | D-48 |
| StationParameter object | D-48 |
| SwitchGatewayService object | D-49 |
| StationTypes object | D-49 |
| SwitchService object..... | D-50 |
| System object | D-50 |
| SystemSetting object..... | D-51 |
| SystemTarget object..... | D-52 |
| SystemSettings object | D-52 |
| User object..... | D-53 |
| View object | D-55 |
| Views object..... | D-56 |
| Workgroup object..... | D-56 |
| WorkgroupMember object | D-57 |
| WorkgroupMembers object..... | D-57 |

Appendix E. Client API Enumerations

| | |
|--------------------------------|-----|
| TVAddressCategory | E-1 |
| TVAddressError | E-1 |
| TVAddressFieldValidation | E-1 |
| TVAddressPhoneComponent | E-2 |
| TVAddressSubType..... | E-2 |
| TVAddressType | E-2 |
| TVAddressUsageType..... | E-2 |
| TVAgentStatisticInterval | E-2 |
| TVAgentStatus..... | E-3 |
| TVApplicationType..... | E-3 |
| TVAssociatedPersonType | E-3 |

| | |
|-------------------------------------|------|
| TVAudioDeviceType | E-3 |
| TVAudioState..... | E-3 |
| TVAudioStateChangeReason..... | E-4 |
| TVAudioType | E-4 |
| TVAudioVOXFormat | E-4 |
| TVCallAnnounceType..... | E-4 |
| TVCallDeclineMode | E-4 |
| TVCallDirection..... | E-4 |
| TVCallerIDFormat | E-4 |
| TVCallFeature..... | E-5 |
| TVCallFieldValidation | E-5 |
| TVCallGrabAndHoldAudioType | E-5 |
| TVCallHistoryError..... | E-6 |
| TVCallHistoryFieldValidation | E-6 |
| TVCallHistoryResult..... | E-6 |
| TVCallRecordFormat | E-6 |
| TVCallRecordStatus | E-6 |
| TVCallRuleCallerTypeCondition | E-6 |
| TVCallRuleError..... | E-7 |
| TVCallRuleFieldValidation | E-7 |
| TVCallRuleRingOverride | E-7 |
| TVCallRuleScheduleType..... | E-7 |
| TVCallType | E-7 |
| TVColumnsError | E-7 |
| TVContactAssociateFlags | E-8 |
| TVContactError..... | E-8 |
| TVContactFieldValidation | E-8 |
| TVDefaultFolders..... | E-9 |
| TVDeleteMode..... | E-9 |
| TVDeviceHookState | E-9 |
| TVDirectorySearchMode | E-9 |
| TVError | E-10 |
| TVExportFormatType | E-13 |
| TVExportType..... | E-13 |
| TVExtensionDialByNameAction | E-13 |
| TVFieldDataType..... | E-13 |
| TVFilterOperator..... | E-13 |
| TVFinalActionError | E-14 |
| TVFolderError | E-14 |
| TVFolderFieldValidation | E-14 |
| TVFoldersError | E-14 |
| TVGreetingError | E-14 |
| TVGreetingFieldValidation..... | E-14 |
| TVImportContactField..... | E-15 |
| TVImportHeaderPosition | E-15 |
| TVImportOptions..... | E-15 |
| TVInterface | E-15 |
| TVItemChangedBy | E-16 |
| TVItemStatus | E-16 |
| TVLicenseStatus..... | E-16 |
| TVLicenseType..... | E-16 |
| TVLogCallsLevel..... | E-16 |
| TVMessageAssociatedPersonType..... | E-16 |
| TVMessageError..... | E-17 |

| | |
|---------------------------------------|------|
| TVMessageFieldValidation | E-17 |
| TVMessageStatus | E-17 |
| TVMessageType..... | E-17 |
| TVNameFormat | E-17 |
| TVNotificationError | E-18 |
| TVNotificationFieldValidation..... | E-18 |
| TVNotificationLevel..... | E-18 |
| TVNotificationMessageAction..... | E-18 |
| TVNotificationType | E-18 |
| TVObjectClass..... | E-19 |
| TVObjectDialingServiceClass..... | E-20 |
| TVObjectItemClass..... | E-20 |
| TVOutboundCallerIDPresentation | E-21 |
| TVPartyDirection..... | E-21 |
| TVPartyFeature | E-21 |
| TVPartyHistoryResult | E-21 |
| TVPartyRole | E-21 |
| TVPartyStatus..... | E-22 |
| TVPermissionAccessLevel | E-22 |
| TVPermissionType | E-22 |
| TVPersonalStatusCallForwarding..... | E-22 |
| TVPersonalStatusError..... | E-23 |
| TVPersonalStatusFieldValidation | E-23 |
| TVPersonalStatusType..... | E-23 |
| TVRecipientError | E-23 |
| TVRegistryDataType | E-23 |
| TVRegistryLocation | E-23 |
| TVRoutingListActionError | E-24 |
| TVRoutingListActionType | E-24 |
| TVRoutingListError | E-24 |
| TVRoutingListFieldValidation..... | E-24 |
| TVRoutingListFinalActionType | E-24 |
| TVRoutingListPlayGreetingType | E-24 |
| TVRoutingListPromptCallerType | E-25 |
| TVScheduledDaysWeekday | E-25 |
| TVScheduleItemFieldValidation..... | E-25 |
| TVScheduleItemType | E-25 |
| TVSchedulesError | E-25 |
| TVSearchType..... | E-25 |
| TVServerConnectionLevel..... | E-25 |
| TVServerStatus | E-26 |
| TVSessionStatus | E-26 |
| TVSessionValidateLogonResult | E-26 |
| TVShortcutError..... | E-26 |
| TVShortcutFieldValidation | E-26 |
| TVShortcutGroupFieldValidation | E-26 |
| TVShortcutGroupIconType..... | E-26 |
| TVSortOrder | E-26 |
| TVSpecialUsers..... | E-26 |
| TVStationAnalogPhoneType | E-27 |
| TVStationButtonFieldValidation | E-27 |
| TVStationError | E-27 |
| TVStationFeatureParameterType..... | E-27 |
| TVStationFieldValidation | E-27 |

| | |
|--|------|
| TVStationPadCharacter | E-27 |
| TVStationPadExtension | E-28 |
| TVStationPhoneFeatures | E-28 |
| TVStationTransferMode | E-28 |
| TVStationUsage | E-28 |
| TVStoreHistoryType | E-28 |
| TVSystemTargetAvailability | E-28 |
| TVSystemTargetCallForwardingType | E-28 |
| TVSystemTargetError | E-28 |
| TVSystemTargetType | E-29 |
| TVUserAccountCodeBehavior | E-29 |
| TVUserAnnounceCallBehavior | E-29 |
| TVUserCallWaiting | E-29 |
| TVUserCategory | E-29 |
| TVUserDefaultContactAction | E-29 |
| TVUserEmptyDeletedBehavior | E-29 |
| TVUserError | E-30 |
| TVUserExternalCallTypes | E-30 |
| TVUserFieldValidation | E-30 |
| TVUserInboundCallBehavior | E-31 |
| TVUserMessageOrder | E-31 |
| TVUserNameFormat | E-31 |
| TVUserOutboundCallBehavior | E-31 |
| TVUserPromptCallerForName | E-31 |
| TVUserRingPattern | E-31 |
| TVUserSendDigitsBehavior | E-31 |
| TVUserStrataRingPattern | E-32 |
| TVViewError | E-32 |
| TVViewFieldValidation | E-32 |
| TVWorkgroupErrors | E-32 |
| TVWorkgroupFieldValidation | E-32 |
| TVWorkgroupMembersError | E-32 |
| TVWorkgroupRoutingType | E-32 |

CHAPTER 1

INTRODUCTION

CHAPTER CONTENTS

| | |
|-----------------------------------|-----|
| About the TeleVantage SDK | 1-2 |
| Development tools and terms. | 1-3 |
| Installing the APIs | 1-3 |

About the TeleVantage SDK

The TeleVantage Software Development Kit (SDK) is a collection of COM-based components and interfaces that can be incorporated into your custom applications. See the following chapters for detailed descriptions of the TeleVantage SDK application programming interfaces (APIs):

- **Chapter 2, “The Client API”**, describes the API used to write the TeleVantage Client. The Client API is a set of software components that gives custom applications the ability to access all functions found in the TeleVantage Client. This chapter also provides an example of a simple program that uses the API to retrieve and display information from the TeleVantage Server.
- **Chapter 3, “The IVR Plug-in API”**, describes features that tightly integrate a custom application with the TeleVantage Server to perform basic call handling or voice processing tasks such as order entry, customer service, e-mail readers, and so forth. The application (called an IVR Plug-in) is a “virtual extension” on the TeleVantage Server. The IVR Plug-in can be dialed from a phone or auto attendant, called from an internet trunk, or have calls forwarded or transferred to it, just like a regular extension assigned to a user.
- **Chapter 4, “The Device Status API”**, describes features that your application can use to monitor the status of all devices on the TeleVantage Server. For example, it could monitor current users on the system, obtain the name of a user currently logged in at a station, or identify the trunk to which a station is connected. The application could generate custom reports concerning the calls handled by TeleVantage.

In some cases, the TeleVantage SDK may not be the simplest way to implement a specific function. See the following chapter for some alternatives:

- **Chapter 5, “Using In-band Signaling”**, discusses methods for extending TeleVantage using in-band signaling.

Technical Support for the TeleVantage SDK

Using your web browser you can access the TeleVantage SDK webboard at <http://webboard.artisoft.com/~SDK> to ask TeleVantage programming questions, learn how others did similar programming tasks, download sample projects, and check for any updates to this documentation. You can receive feedback from the entire TeleVantage community—users, TeleVantage consultants, VARs, and Artisoft developers. The TeleVantage SDK webboard can only be used to post TeleVantage questions regarding software development with the TeleVantage SDK. TeleVantage questions that do not involve software development and the TeleVantage SDK should be directed to your TeleVantage provider.

Important: This forum is unofficially moderated by Artisoft employees. That is, Artisoft will attempt to answer most questions within a 24 hour period, but Artisoft can not guarantee that all questions will be answered. Artisoft also can not guarantee the quality of the answers from any non-Artisoft conference participants.

Development tools and terms

Extensions to TeleVantage can be developed using most COM-compliant Windows programming tools, such as Visual Basic, Visual C++, Delphi, and so forth. The following terms will be used frequently in this guide:

- **COM** (Component Object Model) is an object-oriented programming convention that allows different software components to work together as a single application, even when each component is written without any internal information about the others. The TeleVantage Client API is a collection of COM-based components.
- A **component** is a COM programming structure containing code and data that can be accessed only through a specific set of methods and properties. A component consists of a class and one or more interfaces, defined as follows:
 - A **class** is the program code that determines what the component can do. The class implements a component's methods and properties.
 - An **interface** defines the set of class methods and properties that can be accessed by an application containing the component.
- An **object** is an instance of a class within a running program. For example, the buttons in a program's graphical interface could be objects created as instances of a button class.
- A **module** is a .DLL or .EXE file containing one or more components.
- **Type libraries** contain descriptions of component interfaces and properties, which can be viewed with object browsers such as the one included in Visual Basic. Type libraries can be part of a module, or they can be separate files with a .TLB extension.
- An **API** (Application Programming Interface) is a package of components, type libraries, and other tools that provide your application with a relatively simple way to access some of the functionality in complex software such as TeleVantage.

Installing the APIs

Installing the Device Status API and IVR Plug-in API

The IVR Plug-in API is automatically installed on your TeleVantage server. The TeleVantage SDK installation installs the Device Status API, IVR Plug-in API, and sample applications to the directory you specify. The Device Status API requires that the PC also have the TeleVantage Client, Administrator or Device Monitor installed. While you can develop IVR Plug-in applications on any PC, you can only test and run them on a TeleVantage Server. Note that installing the SDK on your TeleVantage Server may require a restart of the TeleVantage Server, so you should only perform an installation at a time when you can shut the Server down without interrupting users.

To install the TeleVantage SDK

1. Run the TVSDK.exe program located in \TeleVantage on the TeleVantage CD.
2. Follow the instructions in the TeleVantage SDK Setup window.

Installing the Client API

When you install the TeleVantage Client, the Client API is installed automatically. Your development environment must then be configured to use the API. This process is described for Visual Basic in "Starting a new Client API project in Visual Basic" on page 2-20. If you are not using Visual Basic, see your environment's documentation for instructions on adding COM components or type libraries.

CHAPTER 2

THE CLIENT API

CHAPTER CONTENTS

| | |
|---|-------|
| About the TeleVantage Client API | .2-2 |
| Client API object structures | .2-2 |
| Connecting to the Client API | .2-3 |
| Structure of a session | .2-4 |
| The System object | .2-5 |
| The User object | .2-6 |
| How folders are used | .2-7 |
| Other object structures | .2-10 |
| Starting a new Client API project in Visual Basic | .2-20 |
| Programming tips and examples | .2-21 |

About the TeleVantage Client API

The Client API contains the core objects used to create the TeleVantage Client, and makes it possible for your application to do anything that the Client can do. To develop your application, you can use any development environment that uses standard Windows COM components, for example Visual Basic, Visual Basic for Applications (VBA), VBScript, JavaScript, C++, C#, Delphi, and so forth. You can extend existing applications such as Outlook, Goldmine, Seibel, and Onyx, and you can write applications that will run on a web page.

Here are some examples of things you can create or embed in other applications:

- Your own Client GUI. Your application could do everything the TeleVantage Client does, or you could choose a subset of Client functions that fit your specialized needs.
- A specialized operator's console designed to maximize an operator's ability to supervise the phone system.
- A GUI representing a TeleVantage phone, that looks and operates just like a real phone.
- An application to display a list of people who are currently available to answer calls.
- A small application to change your greeting every day, for friends, or for everyone.
- An application that performs wake up calls or reminder calls at scheduled times.
- A call tracking application that sends you notification of every call made and makes it easy to track calls in a way that fits your specialized needs.
- An application to synchronize your TeleVantage contacts with contact databases.
- A small utility that resides in the system tray and that allows you to change you TeleVantage personal status.
- A small, lightweight, skinable call monitor that resides in the system tray.
- A voice mail export application. The API can collect your voice mail and export it in WAV format. Your application can then treat your messages just like any other WAV files, for example, archiving them, storing them in a database, or even converting them to MP3 format and downloading them to your MP3 player or PDA.

Client API object structures

Objects in the Client API are all members of logical groups or object structures. This chapter summarizes the structure and function of these groups.

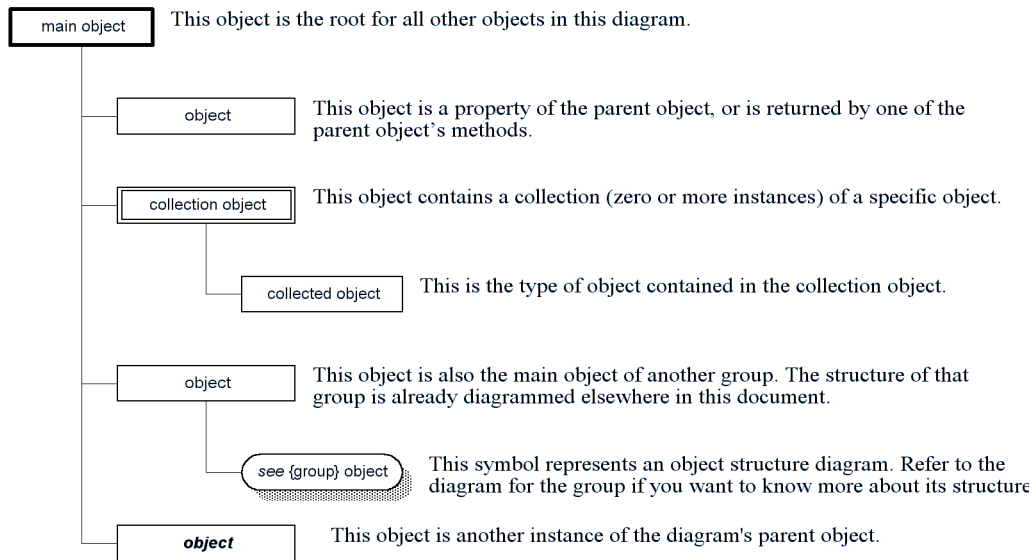
The following sections describe the following top-level Client API object structures:

- **Session.** See "Structure of a session" on page 2-4.
- **System.** See "The System object" on page 2-5.
- **User.** See "The User object" on page 2-6.
- **Folder.** See "How folders are used" on page 2-7.

All of the other major object structures used by the Client API are described in "Other object structures" starting on page 2-10.

Key to object group diagrams

The discussion of each group is accompanied by a diagram that uses the following conventions:



Connecting to the Client API

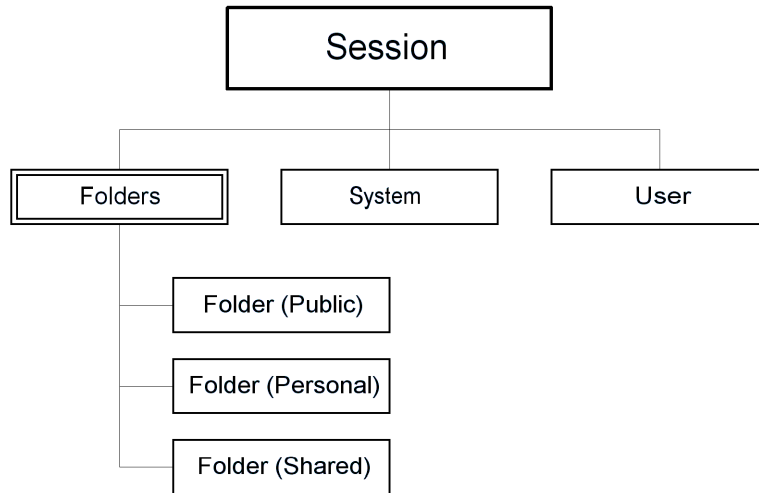
You connect to the Client API on a per-user basis by logging in with a valid user name and password. To be able to view information belonging to more than one user, do either of the following:

- Use sharing to view the other user's voice mailbox, Call Log, Call Monitor, and so forth. For information about sharing, see *Administering TeleVantage*.
- Log on to the Client API multiple times by creating multiple Session objects. You must provide a valid user name and password for each session. You cannot bypass entering a password when logging on to the Client API.

Structure of a session

The Session object is the root of the Client API. It contains methods for logging on and off a TeleVantage Server, and its properties include System and User objects. When the Session logs on to a Server, the System and User objects are automatically populated with information about the current system and the Session's user profile.

All other information is accessed from the Session's Folders object. By default, the Folders object contains three Folder objects that cannot be deleted, and these Folder objects have a default Folder structure that is discussed in detail starting on page 2-9.



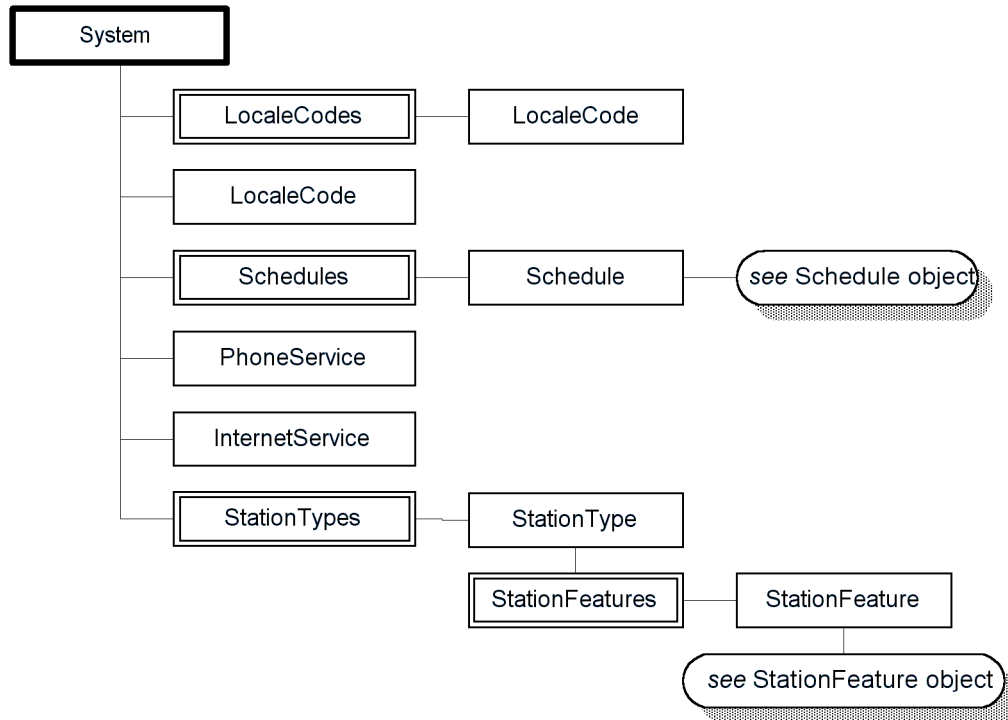
This structure uses the following objects:

- **Session object (see page D-38).**
- **Folders object (see page D-15).** A collection of Folder objects. See “Folder types” on page 2-10 for a description of the various folder types.
- **System object (see page D-50).** Represents the system as a whole. See “The System object” on page 2-5 for a detailed description of System object structure.
- **User object (see page D-53).** Represents the currently logged in end user. See “The User object” on page 2-6 for a detailed description of User object structure.

The System object

The System object represents the system as a whole. It serves as the main interface between the Client and the Server. It contains all system-wide settings and licenses, and handles all operations that affect the system as a whole. Operations performed through the System object will affect the entire system. Typically, only Administrator operations will use the System object.

The System object has the structure shown below:



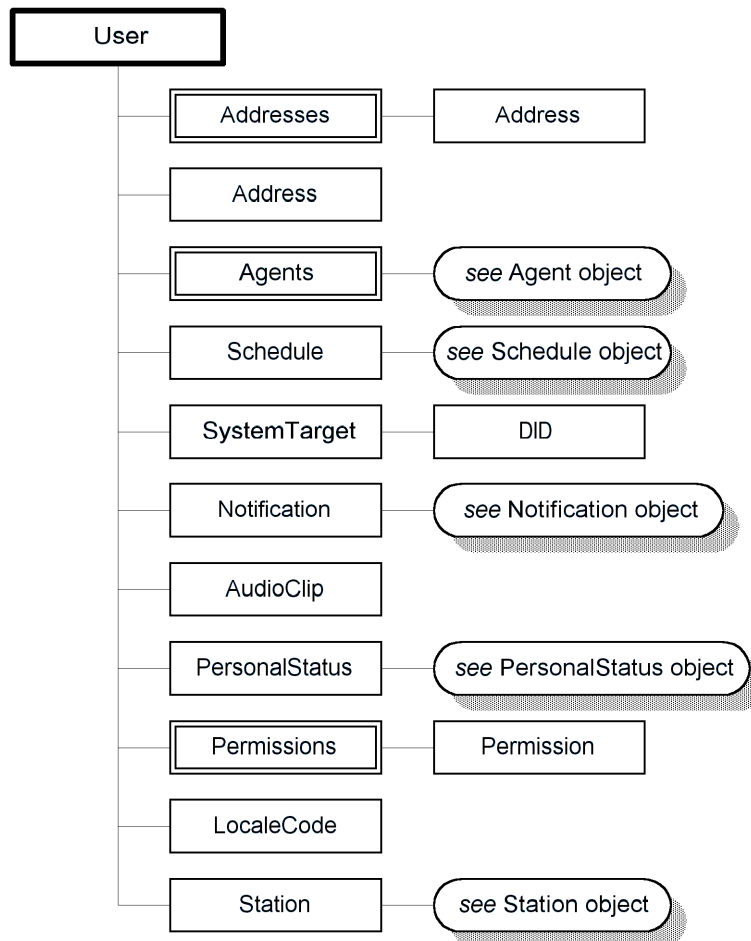
This structure uses the following objects:

- **LocaleCodes object (see page D-20).** A collection of LocaleCode objects. Used only by the System object.
- **LocaleCode object (see page D-19).** Contains a locale code identifying a TeleVantage language locale setting.
- **Schedules object (see page D-38).** A collection of Schedule objects. Used only by the System object.
- **Schedule object (see page D-36).** Contains a record of the times when pager and e-mail message notifications should be sent. See “Schedule objects” on page 2-18 for a detailed description of Schedule object structure.
- **PhoneService object (see page D-28).** A dialing service object containing information for dialing a number over a standard phone line. See “Dialing Service objects” on page 2-14 for a list of all dialing service objects.
- **InternetService object (see page D-18).** A dialing service object containing information for dialing a number over the Internet. See “Dialing Service objects” on page 2-14 for a list of all dialing service objects.
- **StationTypes object (see page D-49).** A collection of StationType objects. Used only by the System object.
- **StationType object (see page D-48).** Contains a list of features used by a specific type of feature phone.
- **StationFeatures object (see page D-47).** A collection of StationFeature objects. Used only by the StationType object.
- **StationFeature object (see page D-47).** Contains information about a feature assigned to a StationButton object. See “StationFeature objects” on page 2-19 for a detailed description of StationFeature object structure.

The User object

The Session's User object represents the currently logged in end user. Depending on Permission settings, a Session User may have permission to modify its own settings, but may still be restricted from reading or modifying the settings of other Users. Two variations of User have been introduced to resolve this conflict — Normal and Current. The difference between the two variations lies in how properties are read and written, and how methods are executed. Both User variations may be in use in any given instance of a Session object. The Session will always have a Current User object. There also may be Folders farther down in the Session structure that contain Normal Users.

The User object has the structure shown below:



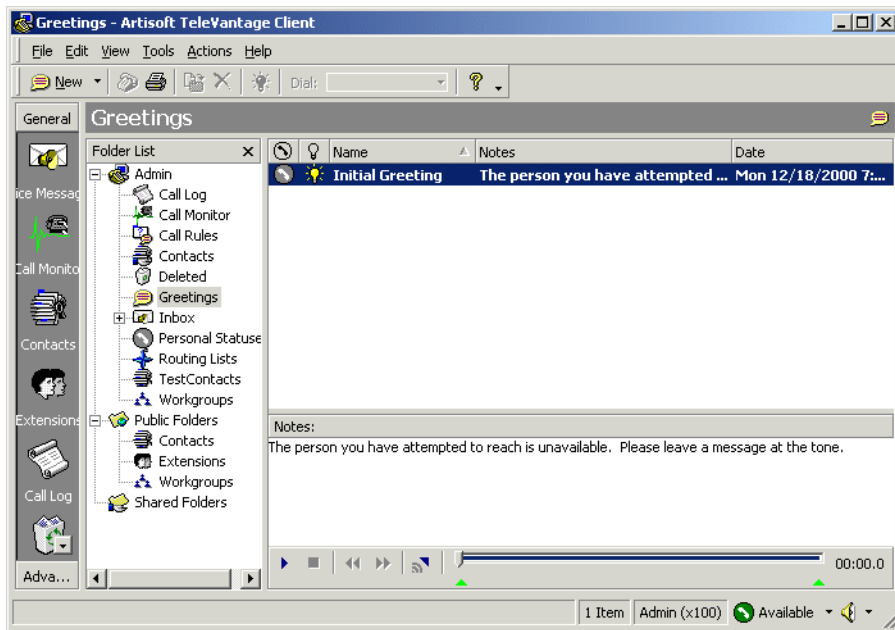
This structure uses the following objects:

- **Addresses object (see page D-3).** A collection of Address objects.
- **Address object (see page D-3).** Contains address information.
- **Agents object (see page D-5).** A collection of Agent objects. Used only by the User object.
- **Agent object (see page D-4).** Contains information about a Call Center agent. See “Agent objects” on page 2-11 for a detailed description of Agent object structure.
- **Schedule object (see page D-36).** Contains a record of the times when pager and e-mail message notifications should be sent. See “Schedule objects” on page 2-18 for a detailed description of Schedule object structure.
- **SystemTarget object (see page D-52).**

- **Notification object (see page D-21).** Contains address and schedule information for sending pager or e-mail message notifications to the user. See “Notification objects” on page 2-16 for a detailed description of Notification object structure. Used only by the User object.
- **AudioClip object (see page D-6).** Used to create or manipulate an audio file.
- **PersonalStatus object (see page D-26).** Contains a record of a personal status, as displayed in the Client’s Personal Status view. See “PersonalStatus objects” on page 2-16 for a detailed description of PersonalStatus object structure.
- **Permissions object (see page D-25).** A collection of Permission objects. The Agent and User objects use this collection to define permissions granted to a specific agent or user. A Folder object contains a collection of the permissions required to access the folder.
- **Permission object (see page D-25).** Contains a record of a permission needed for a specific feature.
- **LocaleCode object (see page D-19).** Contains a locale code identifying a TeleVantage language locale setting.
- **Station object (see page D-45).** Contains configuration information about a station. See “Station objects” on page 2-19 for a detailed description of Station object structure.

How folders are used

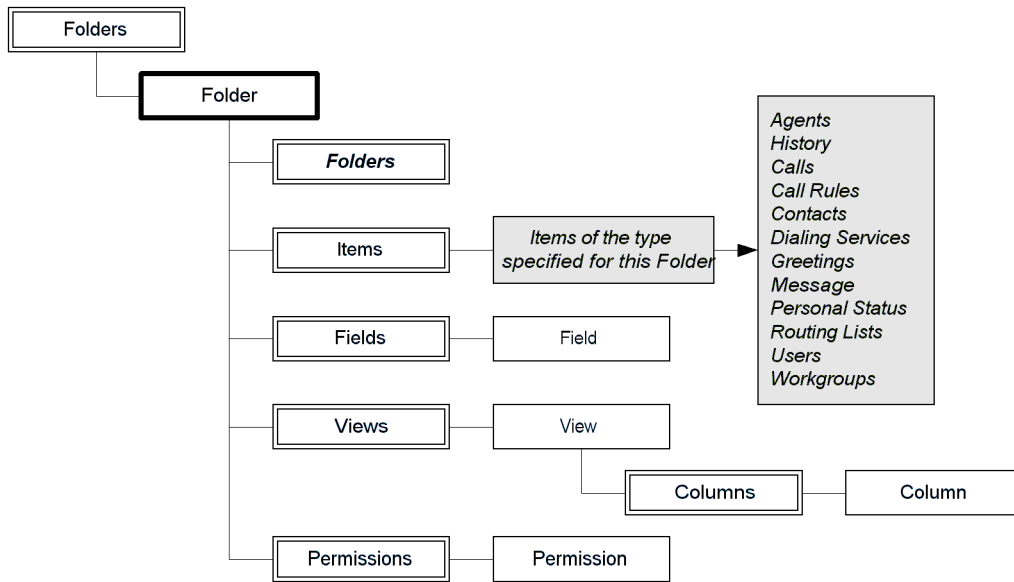
The Client API’s Folder structure corresponds to the structure displayed by the TeleVantage Client. For example, the following illustration shows the TeleVantage Client’s Greetings View.



Each folder used by the Client contains a certain type of information, and has a View that displays the information in rows and columns. The Client API encapsulates such Views in a Folder object structure.

The Folder object

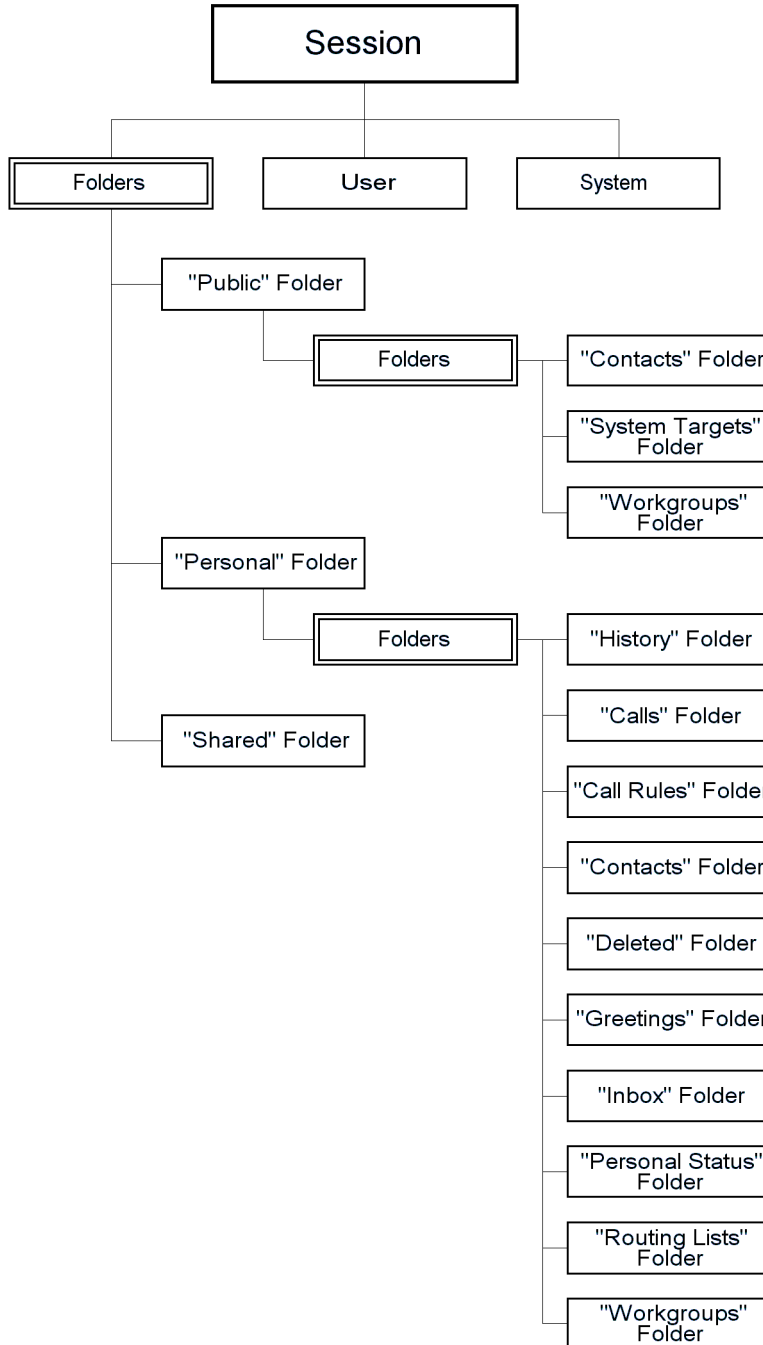
A Folder object contains the following collections of objects:



- **Folders.** Each Folder object contains a Folders collection object, which in turn can contain more Folder objects. Your program's Folder structure can contain as many levels as you wish.
- **Items.** A collection of some specific type of object. For example, the Greetings folder in the example above would contain a set of Greeting objects. Each Greeting object would contain all of the information required for a specific greeting. For a list of Folder types, see "Folder types" on page 2-10.
- **Views, Columns, and Fields.** These collections provide you with information that you can use if you want to duplicate some of the predefined views used by the TeleVantage Client. Your program's user interface does not need to use these collections.
- **Permissions.** A collection of Permission objects that specify how this Folder can be accessed.

Default Folder structure

The Folder objects diagrammed below are always present after a Session object logs on to a Server. They are system defaults, and they cannot be deleted.



Folder types

To access a default folder, you call the Session object's `GetDefaultFolder` method with the appropriate folder constant as the parameter (for example, `AgentFolder = aSession.GetDefaultFolder(tvFolderAgents)`). Valid default folder constants are specified in the enumeration "TVDefaultFolders" on page E-9. The following default folders are available:

| Folder name | Folder constant | Object collected (object structure description) |
|------------------|-------------------------------------|---|
| Agents | <code>tvFolderAgents</code> | Agent object (see page 2-11) |
| Agent Statistics | <code>tvFolderAgentStats</code> | AgentStat object |
| History | <code>tvFolderCallHistory</code> | CallHistory object (see page 2-12) |
| Call Rules | <code>tvFolderCallRules</code> | CallRule object (see page 2-13) |
| Calls | <code>tvFolderCalls</code> | Call object (see page 2-12) |
| Contacts | <code>tvFolderContacts</code> | Contact object (see page 2-13) |
| Current User | <code>tvFolderCurrentUser</code> | Empty root-level folder. Only used to access a user's personal folders. |
| Deleted | <code>tvFolderDeleted</code> | Message object that has been deleted by the user. |
| Greetings | <code>tvFolderGreetings</code> | Greeting object (see page 2-15) |
| Message | <code>tvFolderMessages</code> | Message object (see page 2-15) |
| Personal Status | <code>tvFolderPersonalStatus</code> | PersonalStatus object (see page 2-16) |
| Queue Statistics | <code>tvFolderQueueStats</code> | QueueStat object |
| Routing Lists | <code>tvFolderRoutingLists</code> | RoutingList object (see page 2-17) |
| Services | <code>tvFolderServices</code> | all "Dialing Service" objects (see page 2-14) |
| System Targets | <code>tvFolderSystemTarget</code> | |
| Users | <code>tvFolderUsers</code> | User object (see page 2-6) |
| Workgroups | <code>tvFolderWorkgroups</code> | Workgroup object (see page 2-20) |

Other object structures

Previous sections in this chapter describe the following top-level Client API object structures:

- **Session.** ("Structure of a session" on page 2-4)
- **System.** ("The System object" on page 2-5)
- **User.** ("The User object" on page 2-6)
- **Folder.** ("How folders are used" on page 2-7)

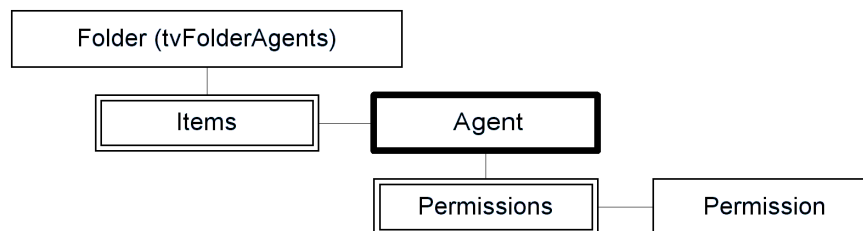
This section discusses all of the other major object structures used by the Client API. Most of these structures have their own Folder type (as described in "Folder types" on page 2-10). The following structures are described:

- **Agent object (see page 2-11).** Contains information about a Call Center agent. Used only in the User object structure.
- **Call object (see page 2-12).** Contains information about a call and methods to manipulate the call, as displayed in the TeleVantage Client Call Monitor view. Created by the Session object when a call is received or placed.
- **CallHistory object (see page 2-12).** Contains information about a call, similar to what is displayed in the Client Call Log view.
- **CallRule object (see page 2-13).** Contains information about a call rule, similar to what is displayed in the Client's Call Rules view.

- **Contact object (see page 2-13).** Contains information about a contact, similar to what is displayed in the Client's Contacts view.
- **Dialing Service objects (see page 2-14).** Contains information about the dialing services defined for a user.
- **Greeting object (see page 2-15).** Contains information about a greeting, similar to what is displayed in the Client's Greetings view. Used in the CallRule, PersonalStatus, and RoutingList object structures.
- **Message object (see page 2-15).** Contains a voice message and related information.
- **Notification object (see page 2-16).** Contains address and schedule information for sending pager or e-mail message notifications to the user. Used only in the User object.
- **PersonalStatus object (see page 2-16).** Contains a record of a personal status, as displayed in the Client's Personal Status view. Used in the CallRule and User objects.
- **RoutingList object (see page 2-17).** Contains information about a routing list, similar to what is displayed in the Client's Routing Lists view. Used in the CallRule and PersonalStatus objects.
- **Schedule object (see page 2-18).** Contains a record of the times when pager and e-mail message notifications should be sent. Used in the CallRule, Notification, System, and User object structures.
- **Shortcut object (see page 2-18).** Contains information that can be used to locate and access a specific Folder object.
- **Station object (see page 2-19).** Contains configuration information about a station. Used in the Session and User objects.
- **StationFeature object (see page 2-19).** Contains information about a feature assigned to a StationButton object. Used in the Station and System object structures.
- **Workgroup object (see page 2-20).** Contains information about a workgroup, similar to what is displayed in the Client's Workgroups view.

Agent objects

Agent objects contain information about a Call Center agent.

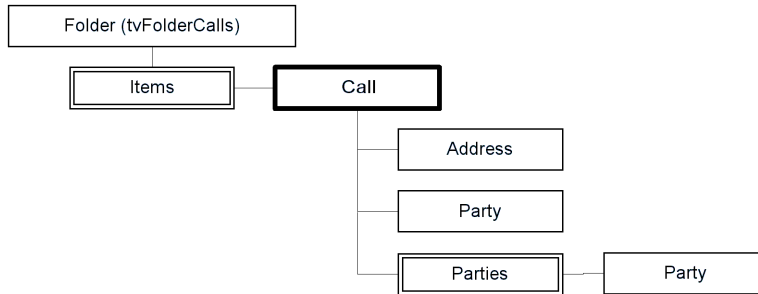


Agent objects can be collected in an Agents object or in a folder of type tvFolderAgents. This structure uses the following objects:

- **Agent object (see page D-4).**
- **Agents object (see page D-5).** A collection of Agent objects. Used only by the User object.
- **Permission object (see page D-25).** Contains a record of a permission needed to access a specific folder. Permission objects can be collected in a Permissions object. Used only by the Permissions object.
- **Permissions object (see page D-25).** A collection of Permission objects. The Agent and User objects use this collection to define permissions granted to a specific agent or user. A Folder object contains a collection of the permissions required to access the folder.

Call objects

Contains information about a call and methods to manipulate the call, as displayed in the Client's Call Monitor view. Created by the Session object when a call is received.

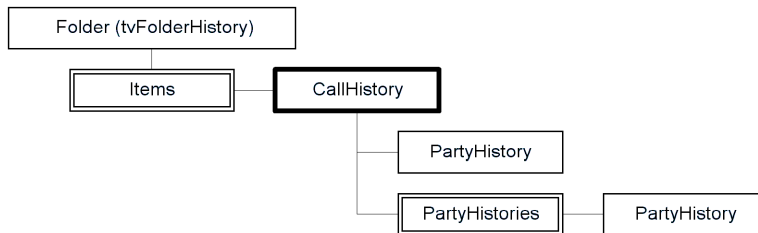


Call objects can be collected in a folder of type tvFolderCalls. This structure uses the following objects:

- **Call object (see page D-7).**
- **Address object (see page D-3).** Contains address information for a contact.
- **Party object (see page D-23).** Contains information about one of the parties involved in a call, such as the caller, the recipient, a member of a conference, and so forth. Party objects can be collected in a Parties object..
- **Parties object (see page D-22).** A collection of Party objects.

CallHistory objects

Contains information about a call, similar to what is displayed in the Client Call Log view.

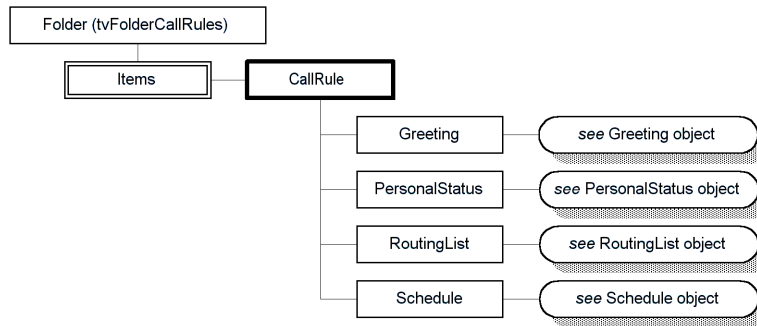


CallHistory objects can be collected in a folder of type tvFolderHistory. This structure uses the following objects:

- **CallHistory object (see page D-9).**
- **PartyHistory object (see page D-24).** Contains information concerning one party in a call.
- **PartyHistories object (see page D-24).** A collection of PartyHistory objects. Used only by the CallHistory object.

CallRule objects

Contains information about a call rule, similar to what is displayed in the Client's Call Rules view.

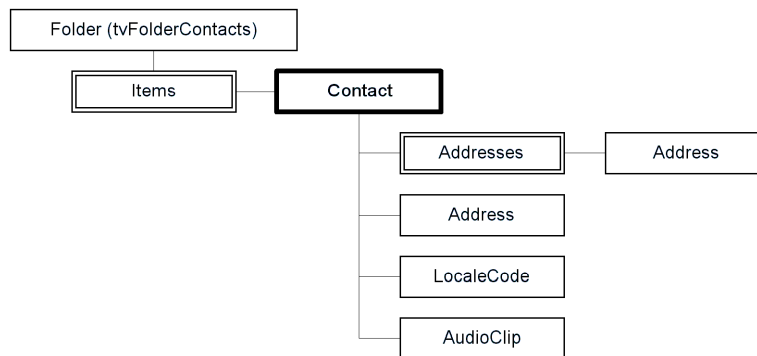


CallRule objects can be collected in a folder of type tvFolderCallRules. This structure uses the following objects:

- **CallRule object (see page D-10).**
- **Greeting object (see page D-15).** Contains information about a greeting, similar to what is displayed in the Client's Greetings view. See "Greeting objects" on page 2-15 for a detailed description of Greeting object structure.
- **PersonalStatus object (see page D-26).** Contains a record of a personal status, as displayed in the Client's Personal Status view. See "PersonalStatus objects" on page 2-16 for a detailed description of PersonalStatus object structure.
- **RoutingList object (see page D-33).** Contains information about a routing list, similar to what is displayed in the Client's Routing Lists view. See "RoutingList objects" on page 2-17 for a detailed description of RoutingList object structure.
- **Schedule object (see page D-36).** Contains a record of the times when pager and e-mail message notifications should be sent. See "Schedule objects" on page 2-18 for a detailed description of Schedule object structure.

Contact objects

Contains information about contacts, similar to what is displayed in the Client's Contacts view. Used by the Address object and the Call, CallHistory, and Message object structures.

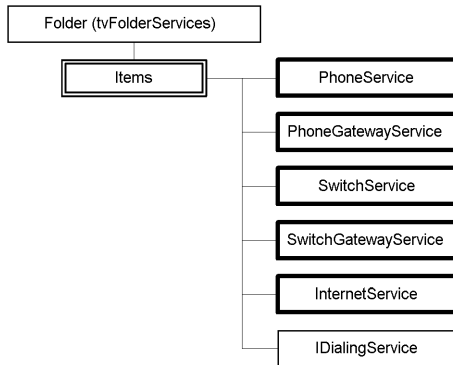


Contact objects can be collected in a folder of type `tvFolderContacts` or in a set of `WorkgroupMembers` objects (see “Workgroup objects” on page 2-20 for a detailed description of Workgroup object structure). This structure uses the following objects:

- **Contact object (see page D-12).**
- **Addresses object (see page D-3).** A collection of Address objects.
- **Address object (see page D-3).** Contains address information for a contact.
- **LocaleCode object (see page D-19).** Contains a locale code identifying a TeleVantage language locale setting.
- **AudioClip object (see page D-6).** Used to create or manipulate an audio file.

Dialing Service objects

Dialing service objects contain information about the dialing services defined for a user. See “Dialing Service types” in *Administering TeleVantage* for a detailed description of each service.

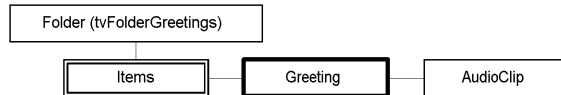


Dialing Service objects are collected in a Folder of type `tvFolderServices`. This structure uses the following objects:

- **InternetService object (see page D-18).** A dialing service object containing information for dialing a number over the Internet.
- **PhoneGatewayService object (see page D-27).** A dialing service object containing information for dialing a number over an IP Gateway.
- **PhoneService object (see page D-28).** A dialing service object containing information for dialing a number over a standard phone line.
- **RoutingService object (see page D-35).** Contains information used to select the dialing service based on the number dialed.
- **SwitchGatewayService object (see page D-49).** A dialing service object containing information for dialing a number over an IP Gateway that connects to a remote Server and then dials the number from that Server.
- **SwitchService object (see page D-50).** A dialing service object containing information for dialing a Centrex or PBX extension or other custom numbers over analog and digital trunks that may be connected to external switches.
- **IDialingService object (see page D-17).** A generic interface used to communicate with any of the dialing service objects.
- **IPhoneService object (see page D-18).** A generic interface used to communicate with PhoneService, PhoneGatewayService, and RoutingService objects.

Greeting objects

Contains information about a greeting, similar to what is displayed in the Client's Greetings view. Used by the CallRule, PersonalStatus, RoutingList object structures.

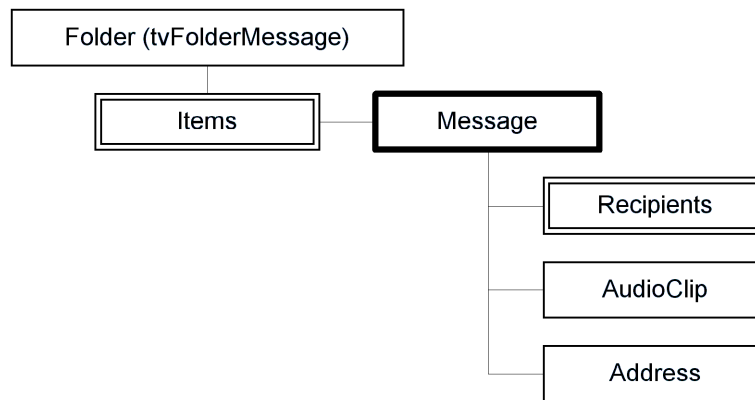


Greeting objects can be collected in a folder of type tvFolderGreetings. This structure uses the following objects:

- **Greeting object (see page D-15).**
- **AudioClip object (see page D-6).** Used to create or manipulate an audio file.

Message objects

Contains a voice message and related information. Used in the CallHistory object structure.

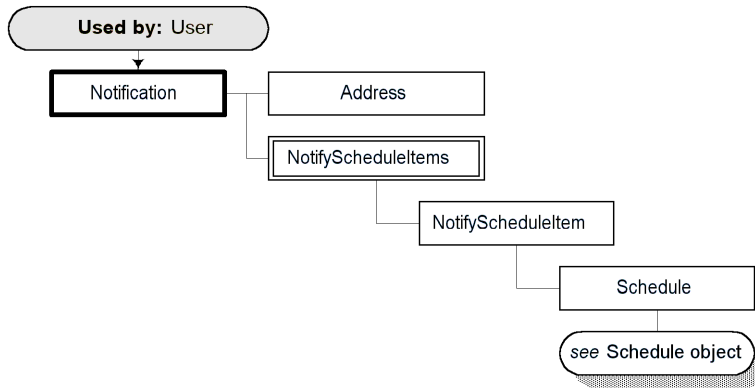


Message objects can be collected in a folder of type tvFolderMessages. This structure uses the following objects:

- **Message object (see page D-20).**
- **Recipients object (see page D-32).** Collection object containing a list of Contact and Workgroup items.
- **AudioClip object (see page D-6).** Used to create or manipulate an audio file.
- **Address object (see page D-3).** Contains address information for a contact.

Notification objects

Contains address and schedule information for sending pager or e-mail message notifications to the user. Used only in the User object.

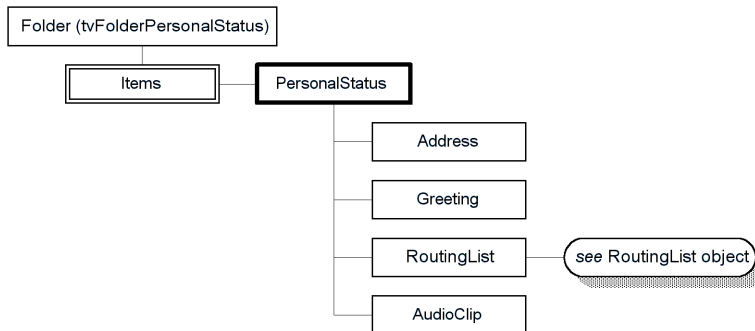


Notification objects are used by the User and Queue objects, but are not collected by any collection object or default Folder type. This structure uses the following objects:

- **Notification object (see page D-21).**
- **Address object (see page D-3).** Contains address information for a contact.
- **NotifyScheduleItems object (see page D-22).** A collection of NotifyScheduleItem objects. Used only by the Notification object.
- **NotifyScheduleItem object (see page D-21).** Contains a record of when pager or e-mail notifications should be sent. Used only by the NotifyScheduleItems object.
- **Schedule object (see page D-36).** Contains a record of the times when pager and e-mail message notifications should be sent. See “Schedule objects” on page 2-18 for a detailed description of Schedule object structure.

PersonalStatus objects

Contains a record of a personal status, as displayed in the Client’s Personal Status view. Used by the CallRule and User objects.

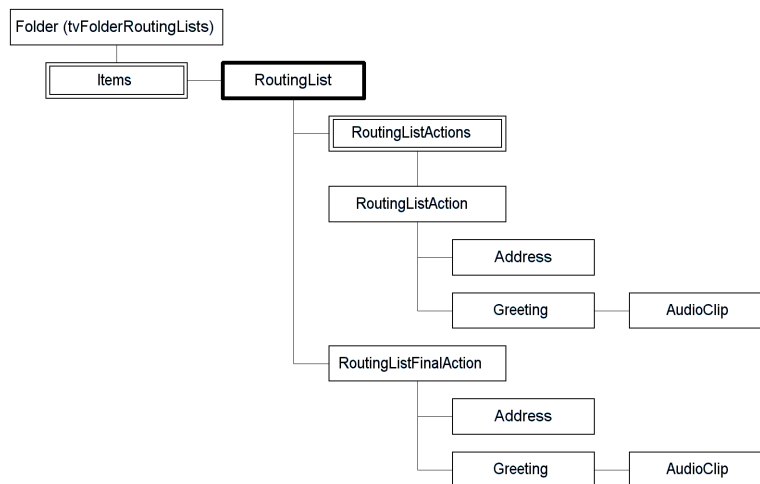


PersonalStatus objects can be collected in a folder of type tvFolderPersonalStatus. This structure uses the following objects:

- **PersonalStatus object (see page D-26).**
- **Address object (see page D-3).** Contains address information for a contact.
- **Greeting object (see page D-15).** Contains information about a greeting, similar to what is displayed in the Client's Greetings view. See "Greeting objects" on page 2-15 for a detailed description of Greeting object structure.
- **RoutingList object (see page D-33).** Contains information about a routing list, similar to what is displayed in the Client's Routing Lists view. See "RoutingList objects" on page 2-17 for a detailed description of RoutingList object structure.
- **AudioClip object (see page D-6).** Used to create or manipulate an audio file.

RoutingList objects

Contains information about a routing list, similar to what is displayed in the Client's Routing Lists view. Used by the CallRule and PersonalStatus objects.

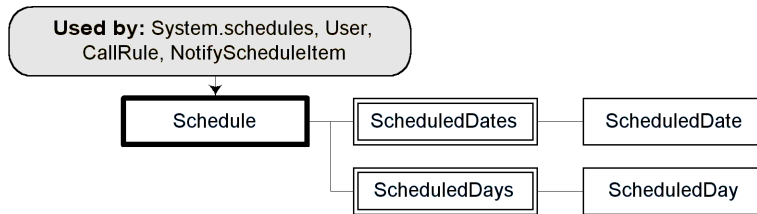


RoutingList objects can be collected in a folder of type tvFolderRoutingLists. This structure uses the following objects:

- **RoutingList object (see page D-33).**
- **RoutingListActions object (see page D-34).** A collection of RoutingListAction objects.
- **RoutingListAction object (see page D-34).** Contains a record of a single routing list action.
- **RoutingListFinalAction object (see page D-35).** Contains a record of the final action in a routing list.
- **Address object (see page D-3).** Contains address information for a contact.
- **Greeting object (see page D-15).** Contains information about a greeting, similar to what is displayed in the Client's Greetings view. See "Greeting objects" on page 2-15 for a detailed description of Greeting object structure.

Schedule objects

Contains a record of the times when pager and e-mail message notifications should be sent. Used by the CallRule, NotifyScheduleItem, and User objects.

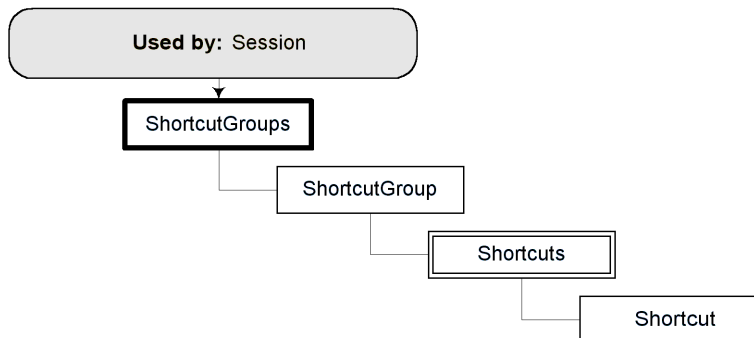


Schedule objects can be collected in the Schedules object. Schedule objects do not have a default Folder type. This structure uses the following objects:

- **Schedule object (see page D-36).**
- **Schedules object (see page D-38).** A collection of Schedule objects. Used only by the System object.
- **ScheduledDates object (see page D-36).** A collection of ScheduledDate objects.
- **ScheduledDate object (see page D-36).** Contains a record of a date and time when pager and e-mail message notifications should be sent.
- **ScheduledDays object (see page D-37).** A collection of ScheduledDay objects.
- **ScheduledDay object (see page D-37).** Contains a record of times on specific days of the week when pager and e-mail message notifications should be sent.

Shortcut objects

Contains information that can be used to locate and access a specific Folder object. Used by the Session objects.

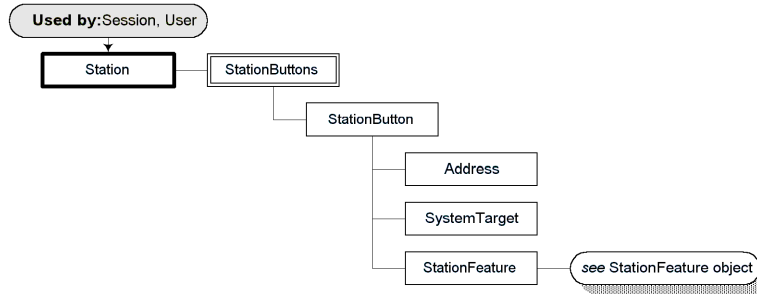


Shortcut objects can be collected in the Shortcuts object. This structure uses the following objects:

- **Shortcut object (see page D-43).**
- **Shortcuts object (see page D-45).** A collection of Shortcut objects.
- **ShortcutGroup object (see page D-44).** Contains information about a group of shortcuts, for example, the General and Advanced shortcut groups displayed in the Client's view bar. ShortcutGroup objects can be collected in the ShortcutGroups object.
- **ShortcutGroups object (see page D-44).** A collection of ShortcutGroup objects.

Station objects

Contains configuration information about a station. Used by the Session and User objects.

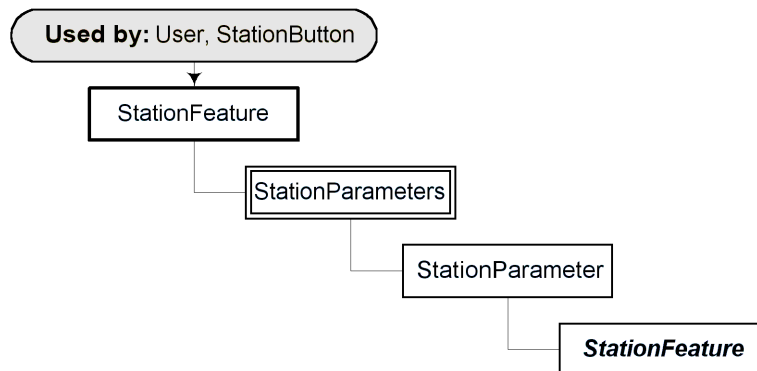


Station objects do not have a default Folder type. This structure uses the following objects:

- **Station object (see page D-45).**
- **StationButtons object (see page D-46).** A collection of StationButton objects.
- **StationButton object (see page D-46).** Contains information about a specialized button on a feature phone.
- **Address object (see page D-3).** Contains address information for a contact.
- **SystemTarget object (see page D-52).**
- **StationFeature object (see page D-47).** Contains information about a feature assigned to a StationButton object. See “StationFeature objects” on page 2-19 for a detailed description of StationFeature object structure.

StationFeature objects

Contains information about a feature assigned to a StationButton object.

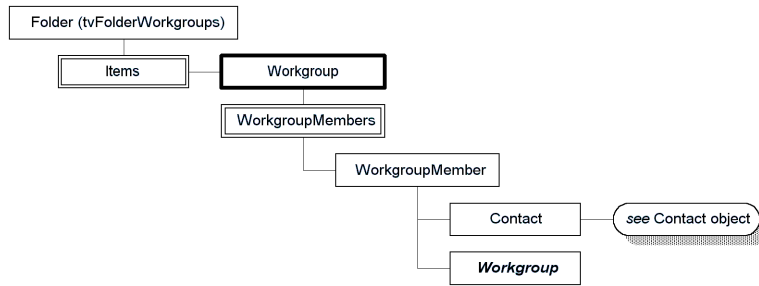


StationFeature objects are used by the User object and are collected in the StationFeatures object (which is used only in the System object). StationFeature objects do not have a default Folder type. This structure uses the following objects:

- **StationFeature object (see page D-47).**
- **StationFeatures object (see page D-47).** A collection of StationFeature objects. Used only by the StationType object in the System object structure.
- **StationParameters object (see page D-48).** A collection of StationParameter objects.
- **StationParameter object (see page D-48).** Contains a parameter required by a StationFeature object.

Workgroup objects

Contains information about a workgroup, similar to what is displayed in the Client's Workgroups view.



Workgroup objects can be collected in a folder of type tvFolderWorkgroups, or in a set of WorkgroupMember objects. Each Workgroup object contains a WorkgroupMembers collection. The WorkgroupMember objects in a WorkgroupMembers collection can contain either Contact objects or Workgroup objects. This structure uses the following objects:

- **Workgroup object (see page D-56).**
- **WorkgroupMembers object (see page D-57).** A collection of WorkgroupMember objects.
- **WorkgroupMember object (see page D-57).** Contains either a Contact object or a Workgroup object.
- **Contact object (see page D-12).** Contains information about contacts, similar to what is displayed in the Client's Contacts view. See "Contact objects" on page 2-13 for a detailed description of Contact object structure.

Starting a new Client API project in Visual Basic

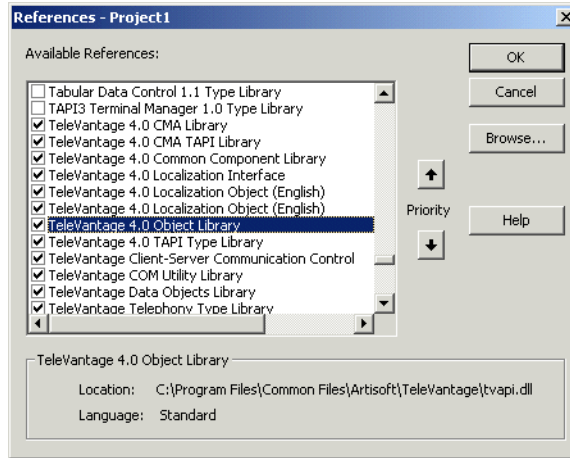
The following procedure creates a Visual Basic project and adds the Client API to it. At that point you can start developing your own Client API application.

To start a new Client API project

1. Start Visual Basic and open a new Standard EXE project.



- In the VB main menu, choose **Project > References**. In the References window, select the checkbox for **TeleVantage 4.0 Object Library**.



Programming tips and examples

This section contains tips and examples that demonstrate how to perform some common tasks via the TeleVantage Client API.

How do I export a list of voice messages?

The following sample code exports a list of voice messages and then plays one over the phone via the API.

' This sample code exports and plays voice messages.

```
Dim oSession As Session
Dim oFileSystemObject As Object
Dim oFile As Object
Dim sExportString As String
Dim oMessageFolder As Folder
Dim oMessageView As View
Dim sMessageID As String
Dim oMessage As Message

Set oSession = New Session

' Log on to the TeleVantage Server.
' "32402" is a unique number used to identify the application using this Session object.
oSession.Logon "Server_Name", "Joe User", "*****", 2, tvApplicationTypeClient, "32402", , False

' Get the Messages Folder.
Set oMessageFolder = oSession.GetDefaultFolder(tvFolderMessages)

' Get the Messages View and export (as XML) the column data.
Set oMessageView = oMessageFolder.Views.Item(1)
sExportString = oMessageView.Export(tvExportTypeData)

' Write the exported data to an XML file.
Set oFileSystemObject = CreateObject("Scripting.FileSystemObject")
Set oFile = oFileSystemObject.CreateTextFile(App.Path & "\ & "MessageData.xml", True)
oFile.write sExportString
oFile.Close
```

```
' Assuming we parsed the XML file for the object ID of a message (and stored it 'in sMessageID)
' Get the actual message object from the system and play it over the handset.
Set oMessage = oSession.GetItem(sMessageID)
oMessage.AudioClip.DeviceType = tvAudioDevicePhone
oMessage.AudioClip.Play
```

How do I create a TeleVantage contact?

```
' This sample code creates a TeleVantage contact.
```

```
Dim oSession As Session
Dim oContactsFolder As Folder
Dim oContact As Contact
Dim oAddress As Address

Set oSession = New Session

' Log on to the TeleVantage Server.
' "32402" is a unique number used to identify the application using this Session object.
oSession.Logon "Server_Name", "Joe Smith", "****", 2, tvApplicationTypeClient, "32402", , False

' Get the Contacts Folder.
Set oContactsFolder = oSession.GetDefaultFolder(tvFolderContacts)

' Create a new Contact object.
Set oContact = oContactsFolder.Items.Add(tvClassContact)

' Set User properties.
oContact.FirstName = "Jim"
oContact.LastName = "Williams"
oContact.PIN = "123"

' Create new Address object.
Set oAddress = oContact.Addresses.Add()

' Set the Address properties for the new Contact.
oAddress.AddressType = tvAddressTypePhoneNumber
oAddress.Category = tvAddressCategoryHome
oAddress.Description = "Jim's home number"
oAddress.Value = "555-1234"
oAddress.Default = True 'this will make this the default address for this contact

' Call the Validate method so that all assigned fields can be checked.
oAddress.Validate (True)

' Save the new Contact.
oContact.Save (True)
```

How do I create a TeleVantage user?

The following sample code creates a TeleVantage user via the API.

```
' This sample code creates a user.

Dim oSession As Session
Dim oUsersFolder As Folder
Dim oUser As User
Dim oAddress As Address

    Set oSession = New Session

' Log on to the TeleVantage Server as a user that has privileges to create users on the Server.
' "32402" is a unique number used to identify the application using this Session object.
oSession.Logon "Server_Name", "Admin", "100", 2, tvApplicationTypeClient, "32402", , False

' Get the Users Folder.
Set oUsersFolder = oSession.GetDefaultFolder(tvFolderUsers)

' Create a new User object.
Set oUser = oUsersFolder.Items.Add(tvClassUser)

' Set User properties.
oUser.FirstName = "Joe"
oUser.LastName = "Smith"
oUser.extension = "117"
oUser.Company = "Joe's Stuff, Inc."
oUser.VoiceTitle.Import "c:\voice titles\JoeVoiceTitle.wav"

' Create a new Address object.
Set oAddress = oUser.Addresses.Add()

' Set the Address properties for the new User.
oAddress.AddressType = tvAddressTypeStation
oAddress.Category = tvAddressCategoryBusiness
oAddress.Description = "Address for Joe Smith"
oAddress.Value = 2 'station 2
oAddress.Default = True 'this will make station 2 Joe's default address

' Call the Validate method so that all assigned fields can be checked for correctness.
oAddress.Validate (True)

' Save the new User.
oUser.Save (True)
```

How do I create a call?

Use `Session.CreateAddress` and `Session.CreateCall` (see “Session object” on page D-38). To create a call to a user, get the user's ID and create an address of type `tvAddressTypeUser`, as shown in the following example.

```
' This sample code creates a call to a user.

Dim oSession As Session
Dim oSystemTargetFolder As Folder
Dim oUser As SystemTarget 'System Targets are items that appear in the extension view
Dim oAddress As Address

Set oSession = New Session

' Log on to the TeleVantage Server.
' "32402" is a unique number used to identify the application using this Session object.
oSession.Logon "Server_Name", "Joe Smith", "****", 2, tvApplicationTypeClient, "32402", , False

' Get the user's ID and set the address type to tvAddressTypeUser.
Set oSystemTargetFolder = oSession.GetDefaultFolder(tvFolderSystemTarget)
Set oUser = oSystemTargetFolder.Items("Sally Jones", tvSearchKey)
Set oAddress = oSession.CreateAddress(oUser.ID, tvAddressTypeUser)
Call oSession.CreateCall(oAddress)
```

To create an external call, you need to use a dialing service. To use a Phone Number dialing service, create an address of type `tvAddressTypePhoneNumber`, as shown in the following example.

```
' Create an external call.

Dim oSession As Session
Dim oAddress As Address
Dim oService As Object

Set oSession = New Session

' Log on to the TeleVantage Server.
oSession.Logon "Server_Name", "Joe Smith", "****", 2, tvApplicationTypeClient, "32402", , False

' Create an Address object that contains the information necessary to make a call.
Set oAddress = oSession.CreateAddress("617-555-1212", tvAddressTypePhoneNumber, , _
                                     oSession.System.DefaultPhoneService)

' Start the call. The station used in the Logon method above should ring.
Call oSession.CreateCall(oAddress)
```

How do I answer an incoming call?

```
' This sample code answers an incoming call.

' Global Variables
Private WithEvents oSession As Session
Private sCurrentCallID As String
Private bCurrentlyInACall As Boolean
Private oCurrentCall As TeleVantage.Call

Private Sub oSession_CallStatusChange(ByVal ID As String, ByVal Status As TeleVantage.TVPartyStatus,
ByVal OldStatus As TeleVantage.TVPartyStatus)

' We get this event anytime the status of any Call object changes.
' The value of ID is a unique identifier for each instance of a Call object.
```

```
' We can use OldStatus and Status to see what the actual transition was.
' For example, you could check to see if a particular call went from Ringing to Active.
' In this very basic example, we will answer a call only if its status is "Ringing" and
' ignore any other incoming calls until the first one hangs up.

    ' Helpful debug statements
    Debug.Print "START oSession_CallStatusChange"
    Debug.Print "Call: " & ID
    Debug.Print "Old Status: " & OldStatus
    Debug.Print "New Status: " & Status

' Since this event can be raised for any number of concurrent calls, check the unique ID to make sure
' we are dealing with the same Call object. Ignore any other call objects for the time being.
    If sCurrentCallID <> ID Then
        Debug.Print "New Call!"
        sCurrentCallID = ID
        ' Get the actual Call object based on the ID.
        Set oCurrentCall = oSession.GetItem(sCurrentCallID)
    Else
        Debug.Print "Existing Call!"
    End If

' Make sure that
' 1) the caller did not hang up between the beginning of this event and now, and
' 2) we are not currently in a call.
    If (oCurrentCall.Status <> tvPartyStatusDisconnected) And (Not bCurrentlyInACall) Then
        ' Answer the call if the status of the Call object is "Ringing".
        If oCurrentCall.Status = tvPartyStatusRinging Then
            oCurrentCall.Answer
            bCurrentlyInACall = True 'ignore any other incoming calls for now
        End If
    End If

' If the call we are in gets disconnected, clear out the ID variable and
' set bProcessingCall to False so we can process the next call.
    If (bCurrentlyInACall) And (oCurrentCall.Status = tvPartyStatusDisconnected) Then
        sCurrentCallID = ""
        bCurrentlyInACall = False 'now we can allow another incoming call
    End If

    Debug.Print "END oSession_CallStatusChange"

End Sub
```

How can I hold and retrieve a call?

Call.Hold puts a call on hold. Call.Answer will take it back (see "Call object" on page D-7).

How can I identify who owns a call?

Call.OwnerParty identifies who owns a call. This is useful when looking at a shared Call Monitor or a call center queue to identify which user TeleVantage is targeting for the call (by ringing that user's phone) or which user has answered.

How do I locate a dialing service by its access code?

The Service folder contains information on all dialing services. You can look at the `AccessCode` property of each service.

```
' This sample code loops through the Service folder and matches the AccessCode.  
Dim oServiceFolder As Folder  
  
Set oServiceFolder = oSession.GetDefaultFolder(tvFolderServices)  
For Each oService In oServiceFolder.Items  
    If (oService.AccessCode = "9") Then  
        Exit For  
    End If  
Next
```

How can I find a specific Call object

When dealing with inbound calls, you can use the `Session` `ItemAdd`, `ItemChange`, `ItemRemove`, or `CallStatusChange` events to get the ID of each call. You can then pass the ID to `Session.GetItem()`, which will return the `Call` object. To determine which folder the call is in, use `Call.Parent`.

The Client can have multiple call folders. In the Client GUI, each call folder maps to the tabs at the bottom of the Call Monitor window (though there is no "all calls" folder). For example, Queue calls that haven't yet been routed to your logged-in user would be in the Queue's call folder. If you know which folder you want to work with, you can loop through the calls in that folder, as shown in the following example.

```
' This sample code loops through a Calls folder so we can work with a specific call.  
Set fCall = oSession.GetDefaultFolder(tvFolderCalls)  
For Each oCall In fCall.Items  
    ' <do something with the call>  
Next
```

How do I check the status of a user?

To check the personal status of the currently signed in user, check `Session.User.LastAppliedStatus`. To check other users' statuses, you need read permission to the `User` folder. Loop through the folder until you find the appropriate user and check the `User.LastAppliedStatus`.

If you are interested in the status of an ACD workgroup agent, check to see if the user is a member of the appropriate workgroup, and check the `User.ACDDoNotDisturb` property.

How do I change personal status?

This example shows how to change personal status via the API. One way to use this technique is to integrate this code with a program written using the Microsoft Outlook API that monitors scheduled events in Outlook. Whenever a scheduled meeting starts, the program automatically changes personal status to "In a Meeting". When the meeting completes, the program then changes personal status back to "Available".

```
' This sample code changes Personal Status to "In a Meeting"  
  
Private WithEvents oSession As Session  
Private Sub Command1_Click()  
  
    ' Session and User object variables  
    Dim oSession As New Session  
    Dim oUser As User  
  
    ' Variables to hold the collection of Personal Status objects for user  
    Dim oPersonalStatusFolder As Folder  
    Dim oPersonalStatus As TeleVantage.PersonalStatus  
  
    ' Log on to the TeleVantage Server.  
    ' "32402" is a unique number used to identify the application using this Session object.
```



```
oSession.Logon "Server_Name", "Joe Smith", "****", 2, tvApplicationTypeClient, "32402", , False

' Get the User object
Set oUser = oSession.User

' Get the Personal Status Folder
Set oPersonalStatusFolder = oSession.GetDefaultFolder(tvFolderPersonalStatus)

' Look at each Personal status object in the folder until we find "*In A Meeting"
' Note that the Name field of statuses are prefixed with "*"
For Each oPersonalStatus In oPersonalStatusFolder.Items
    If oPersonalStatus.Name = "*In A Meeting" Then
        ' Use the ApplyStatus method with our "*In A Meeting" Personal status object
        oUser.ApplyStatus oPersonalStatus
    End If
Next

' Destroy objects and log off Server
Set oUser = Nothing
Set oPersonalStatusFolder = Nothing
Set oPersonalStatus = Nothing
oSession.LogOff

End Sub
```

How can I record conversations between call center agents and external callers?

Call.RecordStart, Call.RecordStop, Call.RecordPause, and Call.RecordResume allow you to record both queue and non-queue calls.

In the following example, Target specifies which user's inbox to which the recording will be sent. The time stamp and name of the person who did the recording are included as well. All recordings are sent to the user's inbox; you cannot send recordings to a different folder. If you do not modify the default Timeout parameter, the recording will not stop until some outside event occurs such as a RecordStop. TermDigits allows you to specify a key (for example, #) that stops the recording when the key is pressed on the keypad.

```
RecordStart(Target, [Format], [Timeout], [TermDigits], [Beep])
Target as SystemTarget object.
Format as TVCallRecordFormat. Optional. Default value is 3 (tvCallRecordFormatPCM8K).
Timeout as Long. Optional. Default value is -1.
TermDigits as String. Optional. Default value is "".
Beep as Boolean. Optional. Default value is 0 (false).
RecordStop()
```

Note: This example uses the default call recording format tvCallRecordFormatPCM8K. If you use a different call recording format, playback of the recording from the TeleVantage Client or converting the recording to a .WAV file in order to forward it via e-mail it will not work. You can use one of the non-default call recording formats if your application manages playback itself.

CHAPTER 3

THE IVR PLUG-IN API

CHAPTER CONTENTS

| | |
|--|-----|
| About the TeleVantage IVR Plug-in API. | 3-2 |
| How the IVR Plug-in components work | 3-3 |
| Using the Customer ID IVR Plug-in | 3-6 |
| IVR Plug-in API quick reference. | 3-9 |

About the TeleVantage IVR Plug-in API

The IVR Plug-in API enables a custom application to function as a “virtual extension” on the TeleVantage Server. The application, called an IVR Plug-in, can act just as if it were using a regular extension assigned to a user.

You can use the IVR Plug-in API to place calls or get notification of new calls from the TeleVantage Server; retrieve Caller ID, DID, or other call data; and then optionally set call data and perform voice processing tasks such as get digits; play or record audio prompts; read numbers, dates, times; perform database lookups, and so forth. After an IVR Plug-in finishes processing the call, it can hang up or transfer the call back to any TeleVantage extension, auto attendant, voice mail box, or even another IVR Plug-in.

IVR Plug-ins must be installed and run on the TeleVantage Server, and you must assign them an extension using the IVR Plug-in view in the TeleVantage Administrator before they can be used.

Important: The TeleVantage IVR Plug-in API provides native support for basic IVR functions (Play, Record, Get Digits, Play Tone, Play Numbers, Play Dollar Amounts, etc.) without requiring a toolkit. If you want to perform advanced IVR functions such as voice recognition, speech synthesis, and advanced call progress analysis you must use the IVR Plug-in API with a voice processing toolkit such as Intel's CallSuite. As of this printing, the IVR Plug-in API is tested to be compatible with CallSuite version 8.2

IVR Plug-in API Component and Sample Applications

The IVR Plug-in API is exposed to your applications through a TeleVantage software component. Sample applications are provided to illustrate how the IVR Plug-in API is used.

- **TVIVRlib type library.** This software component exposes the IVR Plug-in API so your applications can use it to process calls. See “IVR Plug-in API quick reference” on page 3-9 for more information. The library is contained in `tvivr.tlb`, which is located in the `\Program Files\Common Files\Artisoft\TeleVantage` directory when you install the TeleVantage Server or install the TeleVantage SDK.

Several sample IVR Plug-ins written in Visual Basic 6.0 are installed with the TeleVantage SDK. The samples are provided in two versions: `PlugInMedia` samples that use the IVR Plug-in API's built-in voice processing capabilities and `CallSuite` samples that require Intel's Call Suite for voice processing. By default, all samples are located in either the `\Program Files\TeleVantage SDK\PluInMedia` or `\Program Files\TeleVantage SDK\PluInMedia` directories. You must have Visual Basic 6.0 installed to use the sample programs as is, or modify them to meet your needs. The following samples are provided:

- **First.** This small IVR Plug-in demonstrates the most basic layout and design of an IVR Plug-in for a telephony server. It answers a call, plays a file, and then returns the caller to the active routing list.
- **CustID.** This IVR Plug-in demonstrates how to transfer customers to different agents automatically, based on the customer's area code obtained from their Caller ID or their customer record. It also includes a `CustomerID` database manager to maintain the `Customer` and `Agent` database tables accessed by the `CustomerID` IVR Plug-in. The program can be used to customize the customer and agent data for your location.
- **OutBound.** This sample includes two IVR Plug-ins (`PlaceCall` and `ReceiveCall`) that demonstrate how an IVR Plug-in (`PlaceCall`) can place outbound calls that are handled by the `ReceiveCall` Plug-in. It then plays a voice prompt, and then transfer the calls to a TeleVantage user.
- **OrderStatus.** This IVR Plug-in answers calls, prompts the caller for their 5 digit order number, searches a `Orders` database for a matching record and then reads back the order status, e.g. *Your order number [12345] was shipped on [December 31, 2002] and totalled [\$123.56]*

How the IVR Plug-in components work

IVR Plug-ins are, in Microsoft component object model (COM) terms, ActiveX EXEs that you can develop using Visual Basic, Visual C++, Visual Studio .NET or any other Windows development tool that supports the COM requirements of the IVR-plug in interface. Every time the extension of an IVR Plug-in extension is called the TeleVantage Server creates a separate instance of the IVR Plug-in that runs in its own thread of execution (in COM terms an apartment). IVR Plug-ins can then query standard call data such as Caller ID or DID and custom data such as customer ID, and use the facilities provided by Visual Basic or Visual C++ perform any processing such database queries or updates. IVR Plug-ins are network-independent so you do not need to know if the call originated on an IP, T1, E1, analog trunk, or even a phone station. IVR Plug-ins can also update standard or custom call data. All call data follows the call after the IVR Plug-in is finished and can be displayed in the TeleVantage Client's Call Monitor, Call Log and Voice Messages view if properly configured.

IVR Plug-ins can optionally perform voice processing tasks (e.g. *Please enter your order number followed by the # key*) by borrowing a voice resource from the TeleVantage Server. Use the IVR Plug-in GetMedia method to obtain a Media object to perform basic voice processing such as collecting touchtone digits, playing and recording voice files, speaking numbers, money, dates/times, and playing tones. For advanced voice processing that exceed the Media object's capabilities such as voice recognition, speech synthesis, or faxing you must purchase a 3rd party telephony toolkit such as Intel's CallSuite or use the low level Dialogic C API. You then use the IVR-Plug-in GetDevice method to borrow a voice resource from the TeleVantage Server so the 3rd party telephony toolkit can perform voice processing on the call.

When the IVR Plug-in finishes processing the call, can use the IVR Plug-in CallDone method to hang up, transfer to any extension or phone number, or send the call to any voice mail box.

If the caller hangs up before the IVR Plug-in finished processing the call, TeleVantage will notify the IVR Plug-in so it can stop processing and perform any necessary clean up before the IVR Plug-in is terminated. If the IVR Plug-in borrowed a voice resource the TeleVantage Server reclaims the resource at that time. Unresponsive IVR Plug-ins that do not call any IVR Plug-in methods within a 4 minute period (configurable) are also terminated to reclaim voice resources and memory.

Monitoring standard and custom call data

IVR Plug-ins have access to all call basic data such as Caller ID, and can change this call data while processing the call. IVR Plug-ins can also set and change any custom call data you need such as social security number, customer ID, address, and so on. All call data are attached to the party in the call, and if the party is transferred, the call data follows them.

The Client's Call Monitor, Call Log or Voice Messages view can display a list of all the custom call data associated with a call by click **Tools > Columns**, and making sure the **Custom Data** column is displayed.

Additionally the Call Monitor supports adding a unique custom column for each custom data field you want to view:

- To see a custom column in the Call Monitor click **Tools > Columns**, then click **Custom Columns**. Add a custom column with the same name as the key in your IVR Plug-in that you want to display.

Note: You cannot change custom call data from the Call Monitor, but you can from IVR Plug-ins, Auto Attendants, Queues and various TeleVantage add-ons, including the Call Classifier.

IVR Plug-in licensing

IVR Plug-ins are licensed at runtime and count against your station licenses. Your system may be configured to support an unlimited number of IVR Plug-ins, but each simultaneous call to an IVR Plug-in will consume one station license. The station license will be released after the IVR Plug-in calls the CallDone method, or if TeleVantage terminates the IVR Plug-in because it is unresponsive or the caller has hung up.

For example, if you have a system with 16 station licenses, and 15 of those stations are assigned to users, you have one free station license that can be applied to calling an IVR Plug-in. If one person makes a call to an IVR Plug-in, and a second person makes a call to the same or a different IVR Plug-in, the second person will hear a message that the extension is not available, and an error will be logged to the TeleVantage Server's Windows Event Log saying that the station license count has been exceeded.

Note: If you decide to use CallSuite or another voice processing toolkit in conjunction with an IVR Plug-in be aware that these toolkits often impose additional per port licensing fees beyond the station license requirement for any IVR Plug-in. See your telephony toolkit's documentation for details. The IVR Plug-In's PlugInMedia interface performs basic voice processing without requiring additional per-port fees beyond the usual station license requirement.

Registering and Configuring IVR Plug-ins

Before you can use an IVR Plug-in you need to copy and register it on the TeleVantage Server and configure it using the TeleVantage Administrator:

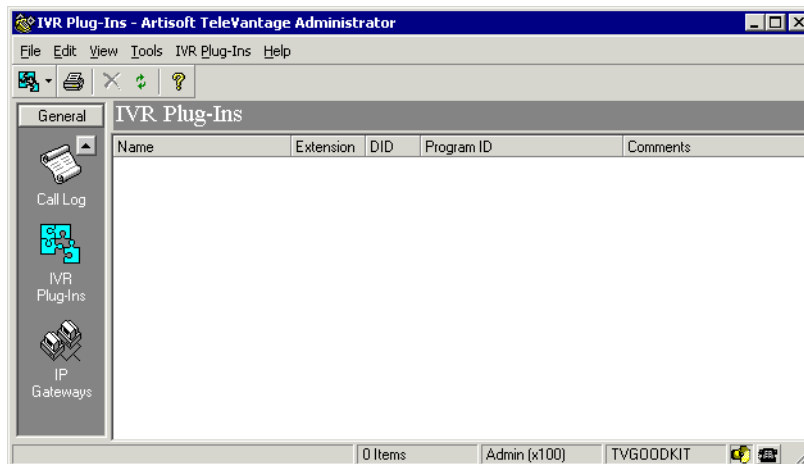
1. If you have compiled your IVR Plug-in to an EXE (using Visual Basic or another tool) on the TeleVantage Server, you can skip to step 3. Otherwise, copy the IVR Plug-in's EXE (e.g. CustID.exe) to any directory on the TeleVantage Server. If applicable, also copy any supporting files required by the IVR Plug-in such as supporting software components or database files.
2. Register your IVR Plug-in's EXE by opening a command prompt and changing to the directory where you copied the IVR Plug-in and entering the following:

```
regsvr32 {IVR Plug-in EXE name}
```

For example:

```
regsvr32.exe custid.exe
```

3. Using the TeleVantage Administrator, switch to the IVR Plug-ins view.




4. Choose **IVR Plug-Ins > New IVR Plug-In**. The IVR Plug-in dialog box opens:

5. Enter the following information:

- **Name.** Name of the IVR Plug-in, for example, “Customer ID”. This name appears in the **Place Call** and **Transfer To** dialog boxes in the TeleVantage Client.
 - **Extension.** Extension assigned to the IVR Plug-in. Any available extension will work. See “Assigning an Extension” in *Administering TeleVantage* for restrictions and recommendations when entering extensions.
 - **DID number.** When TeleVantage recognizes this number as the final digits on an inbound call, the caller is automatically connected to this IVR plug-in, bypassing the main auto attendant. To assign multiple DID numbers to an IVR Plug-in, separate each one with a comma (.).
 - **Program ID.** The program ID of the IVR Plug-in application. For Visual Basic applications, the format is “ActiveXname.Classname”, where ActiveXname is the name of the project (in Visual Basic 6.0 see **Project > Properties**), and Classname is the name of the class that TeleVantage will instantiate. For example, the TeleVantage Customer ID sample application’s program ID is CustID.Medial.
6. To record a voice title for the IVR Plug-in, use the audio controls. The voice title is used in many call-screening features, for example, when an auto attendant transfers a call. The audio controls and how to use them are described in *Administering TeleVantage*.
7. To eliminate startup time when the first call comes in for the IVR Plug-in, select the **Start one instance when the server is started** checkbox.

Note: Startup time depends on the size and complexity of the IVR Plug-in. Most small applications start quickly. If the application takes longer to start, select this option.

8. To initialize a custom data variable each time the IVR Plug-in gets a call, select the **Set custom data when this IVR Plug-in is called** checkbox. Select the **Variable name** from the drop-down list, or click  to create a new variable. Enter the initial **Variable value** in the text box.
Note: The **Organization** checkbox is non-functional and reserved for future use.
9. Click **OK** to save the IVR Plug-in.

Using the Customer ID IVR Plug-in

The IVR Plug-in API includes a sample IVR Plug-in named Customer ID, located in the `\Program Files\TeleVantage SDK\PlugInMedia\CustID` directory. This sample identifies customers via Caller ID and the transfers them to the agent that handles their area code with additional data about the caller so the agent is prepared for the call before they answer.

Specifically the Customer ID IVR Plug-in checks to see if the Caller ID matches a record in a Microsoft Access customer database, and if so searches an agent table for the agent that handles the caller's area code. It then attaches custom data to the call for the customer's city, state and customer ID before transferring the call to appropriate agent.

If there is no matching Caller ID, the Customer ID IVR Plug-in prompts the caller for a Social Security Number (SSN) and matching Personal Identification Number (PIN). If the customer does not have a PIN, they are allowed to enter one. Once identified the customer is handled just as if their Caller ID had been identified. If the sample cannot identify the caller via Caller ID, or SSN/PIN, the call is transferred to the operator.

To run the Customer ID sample:

1. Configure the Customer ID sample as described above in "Registering and Configuring IVR Plug-ins" on page 3-4.
2. Make sure your Server has a user assigned to an extension number that is in the agent database. The sample database already contains an entry for extension number 102. If necessary, you can use the sample database manager to change the extension assignments and customer data in the database (see "Managing the Customer ID database" on page 3-7).
3. Start the TeleVantage Client for the user logged in and assigned to extension 102. To display all the custom data in a single column in the Call Monitor view, choose **View > Current View > Show Columns**, and make sure the **Custom Data** field is added to the list of columns that will be shown.

If you prefer to display each field in separate columns, using the TeleVantage Administrator select **Tools > Custom Data**, click **Add** and specify a string variable called CustomerID, then repeat the process to add another for City, and State. The custom column names are case sensitive, so be sure to enter them exactly as shown. Then you can add those custom columns to the Call Monitor view by choosing **View > Current View > Show Columns**, and add the Customer ID, City and State columns to the list of columns that will be shown.

4. To call the sample IVR Plug-in application, call the extension you assigned to it in the previous steps. If you hear the IVR Plug-in answer the call, skip to step 5.

If you do not hear it answer with an audio prompt, try the following:

- Check the Windows Event Log for any errors generated by the IVR Plug-in
 - Check the Device Monitor for the station or trunk you are using to make the call, to view any custom status messages that the IVR Plug-in is reporting.
 - Make sure you have installed and configured the IVR Plug-in properly as previously described. A common problem is entering the wrong Program ID.
5. Since your call probably does not have a Caller ID that matches any customer record in the sample database, you will be prompted to enter a valid social security number. The sample database already contains a customer record with the social security number **00000000**, which you can enter at the prompt.

6. When prompted, press **5656**, which is the customer's four-digit PIN number.
7. The call should be transferred to extension 102.

The Call Monitor for extension 102 should show the Customer ID, City and State for the customer, either in the Custom Data column or the CustomerID, City, and State columns, depending how you configured your Call Monitor in step 2.

8. Manually transfer the call to another extension. The same custom data will appear in that user's Call Monitor, as long as their Call Monitor is also properly configured to view the custom data. The data also will follow the call to the Call Log and the voice messages view if a message was left.

Managing the Customer ID database

To change or add records to the Customer ID sample IVR Plug-in database, including your own Caller ID numbers and list of agents:

1. Run `CustIDAdmin.exe`, located in the `\Program Files\TeleVantage SDK\PluginMedia\CustID` directory.
2. Change any of the customer data in the Caller Information window.

3. Change any of the agent extension and area code data in the Area Codes window.

Working with the Customer ID sample code

To modify, change or extend the sample application, you must have Microsoft's Visual Basic 6.0 installed, preferably with Service Pack 5 or higher. However, if you just want to view the sample code, Notepad or any text editor will work.

- With Visual Basic 6, open the `CustID.VBP` project located in the `\Program Files\TeleVantage SDK\PluginMedia\CustID` directory. Modify the code as needed and compile the IVR Plug-in to change its behavior.

The Customer ID database program can also be modified using Visual Basic 6 and opening the `CustIDAdmin.VBP` project.

- Without Visual Basic 6, you can see the code and how it works by opening `Media1.cls` in any text editor such as Notepad.

Running an IVR Plug-in that has a GUI

The following steps are required for any IVR Plug-in that has a GUI:

1. On the TeleVantage Server, go to the Windows Control Panel, and select Services. On some versions of Windows, Services is located in the Administrative Tools group.
2. Right-click the TeleVantage Server service and click **Properties**.
3. On the Log On tab, select the **Allow service to interact with desktop** checkbox.
4. Click **OK** to save your change.

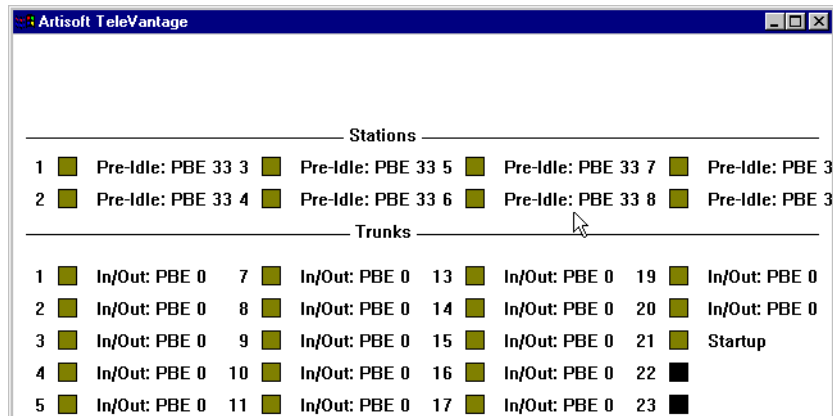
Debugging IVR Plug-ins in Visual Basic

Visual Basic 6.0 cannot debug applications started by Windows services. Since TeleVantage is a Windows service that starts your IVR Plug-in, you need to force TeleVantage to run as a standard executable before you can debug an IVR Plug-in with Visual Basic. These procedures do not have to be performed for IVR Plug-ins written in Visual C++ or other development tools that can debug applications started from Windows services.

To debug an IVR-Plug-in Visual Basic

1. Perform the steps in “Running an IVR Plug-in that has a GUI” on page 3-8.
1. Stop the TeleVantage Server
2. Run RegEdit.
3. Select Edit-Find and find the “TvServ” data
4. Select the LocalService key name (not the TvServ data) and rename it from LocalService to LocalService#. This changes the TeleVantage Server from a Windows Service to a standard executable.
5. Close RegEdit.
6. Start the TeleVantage Server by running TVServer.exe in the \Program Files\TeleVantage Server directory.

The following debugging window will open:



Note: You cannot start the server through the TeleVantage Device Monitor or Administrator when TeleVantage is set up to run as a standard executable rather than a Windows Service. However, you can still run these applications to manage your TeleVantage Server.

7. Open the IVR Plug-in's Visual Basic project and put a break point wherever needed, for example on the line after the CallOffering method.
8. Call your IVR Plug-in's extension. TeleVantage will launch your IVR Plug-in inside Visual Basic and allow you to debug it at the line you set the breakpoint.

When you are finished debugging, you can shut down the TeleVantage Server by closing the TeleVantage Server's debugging window. To reconfigure TeleVantage as a Service, follow steps 1-3 again but this time rename the LocalService# key to LocalService. Now TeleVantage Server will start as a normal Windows Service.

IVR Plug-in API quick reference

This section provides a brief description of each method and property in the IVR Plug-in API. Page numbers refer to the full "IVR Plug-in API Reference" in Appendix A.

Note: For backward compatibility, previous versions of the IVR Plug-in API are supported, and existing IVR Plug-in applications will continue to run. See Appendix B for information on how to migrate existing IVR Plug-in applications to use the current IVR Plug-in API.

PluginCaller interface

| | | |
|------|---------------------------------|--|
| A-2 | AssociateCalledExtension method | Attempts to associate the caller's called ID with a system target based on extension. |
| A-2 | AssociateCalledCallerId method | Attempts to associate the caller's called ID with a personal or public contact. |
| A-3 | AssociateCallerId method | Attempts to associate the Caller ID with a personal or public contact. |
| A-4 | AssociateExtension method | Attempts to associate the Caller ID with a system target based on extension. |
| A-4 | CallDone method | Returns control of the call to the Server with instructions on how it should continue processing the call. |
| A-5 | Delete method | Deletes the specified custom data property from the party in the call. |
| A-5 | DeviceName property | The name of the allocated raw voice resource, if any. |
| A-6 | Exists method | Returns true if the specified property exists. |
| A-6 | Get method | Gets a standard or custom party attribute from the party in the call, for example, ppCallerIDNumber or CustomerID |
| A-8 | GetDevice method | Allocates a raw voice resource from the Server's device pool so that you can pass it to a third party voice processing toolkit. |
| A-9 | GetMedia method | Allocates a voice resource and returns a PluginMedia object associated with it. |
| A-9 | GetXmitTimeslot method | Returns the TDM bus transmit timeslot of the caller's device. |
| A-9 | Listen method | Performs a half duplex route of the media stream, such that the caller's device listens to the specified TDM bus timeslot. |
| A-10 | Ping method | Calling this method tells the server the IVR Plug-in is still running properly. The server will terminate IVR Plug-in's which fail to call this or any PluginCaller method for a period of time. |
| A-10 | PluginComments property | The comments for the called IVR Plug-in as defined in the Administrator. |
| A-10 | PluginDID property | The DID number of the called IVR Plug-in as defined in the Administrator. |
| A-10 | PluginExtension property | The extension of the called IVR Plug-in as defined in the Administrator |
| A-11 | PluginName property | The name of the called IVR Plug-in as defined in the Administrator. |
| A-11 | PluginVariable property | The custom data variable name of the called IVR Plug-in as defined in the Administrator. |
| A-11 | ReleaseDevice method | Returns a raw voice resource to the Server's device pool. |

- A-12 Set method Sets a custom or standard party attribute for the party in the call, for example, CustomerID or ppAccountCode.
- A-13 SetStatus method Sets the status string displayed in the Device Monitor.

PluginApplication interface

- A-14 CallOffering Invoked when the IVR Plug-in's extension is called and the call is offered to the IVR Plug-in.
- A-14 CallPlaced Invoked to notify the IVR Plug-in when an outbound call has been dialed using the PlaceCall method.
- A-14 CallTerminated Invoked when the Server wants to terminate the IVR Plug-in, for example when the server detects that the caller has hung up.

IPluginServer interface

- A-15 PlaceCall method Allows the IVR Plug-in to make an outbound call.

PluginMedia interface

- A-16 Dial method Plays DTMF or MF digits.
- A-17 FlushDigitBuffer method Returns all digits currently in the digit buffer, and leaves the digit buffer empty.
- A-17 GetCallProgress method Performs in-band call progress analysis.
- A-18 GetDigits method Returns a string of digits from the digit buffer based on various criteria including number of digits, time outs and terminating digits.
- A-19 GetXmitTimeslot method Returns the TDM bus transmit timeslot of the voice resource.
- A-19 Listen method Performs a half duplex route such that the voice resource listens to the specified TDM bus timeslot.
- A-20 PlayFile method Plays a pre-recorded audio file.
- A-21 PlayString method Plays a voice string, which is a collection of one or more values played smoothly together using various methods (money, date/time, number, characters, or ordinals).
- A-23 PlayTone method Plays a pre-defined tone such as dial tone or a busy signal.
- A-24 RecordFile method Records a voice file.
- A-25 SetRate method Set audio rate for the next or current playback operation.
- A-25 SetVolume method Set audio volume for the next or current playback operation.
- A-25 Stop method Stops all voice activity.

CHAPTER 4

THE DEVICE STATUS API

CHAPTER CONTENTS

| | |
|---|-----|
| About the TeleVantage Device Status API | 4-2 |
| How the Device Status components work | 4-2 |
| Device Status API quick reference | 4-5 |

About the TeleVantage Device Status API

The Device Status application programming interface (API) is a set of COM components that allow your application to monitor the status of all devices (stations and trunks) on a TeleVantage Server. For example, you could perform any of the following procedures:

- Monitor current users on the system
- Obtain a current list of users permanently assigned to a station
- Obtain the name of a user currently logged in at a station
- Identify which trunk a phone line is connected to

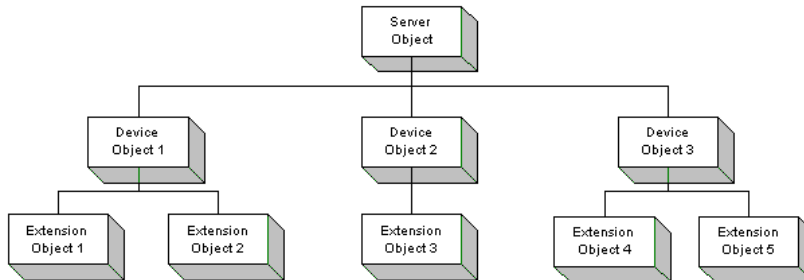
For details about Device Status API methods, properties, and events, see the TeleVantage SDK Help in the directory where you installed the SDK.

The Device Status Sample project

The `TVStatusviewer.vbp` project is a small device monitor application that uses the Device Status components and a standard Visual Basic listview control to display status information.

How the Device Status components work

Device information is organized in a simple hierarchy. In a program that uses the Device Status components, a **Server object** maintains contact with a specified Server and contains a collection of **Device objects** representing all devices on the Server. Each Device object in turn contains a collection of **Extension objects** representing all extensions assigned to that device. The resulting hierarchy might resemble the following diagram:



Server objects

At the top of the hierarchy is the Server object, which uses its **Connect** and **Disconnect** methods to control a connection to a specified TeleVantage Server. Its **DeviceStateChanged** event is raised whenever the state of a device changes. The Server object also contains the **Devices** property, which is a collection of Device objects representing all stations and trunks on the specified Server.

Device objects

Each Device object contains the following read-only properties:

- **Class.** station or trunk
- **HookState.** on-hook, off-hook, or unknown
- **Name.** name of the trunk, or blank if it's a station
- **Number.** positive for stations and negative for trunks
- **Status.** a number indicating the device's current activity

The **Refresh** method updates a Device object's information.

The Device object also contains an Extension object for each extension assigned to the device.

- The **AssignedExtensions** property is a collection that contains one Extension object for each assigned extension. The **CurrentExtension** property is an object representing the extension currently logged in to the device.
- The **DefaultExtension** property is an object representing the device's default extension.

Extension objects

Each Extension object contains the following read-only properties:

- The **DoNotDisturb** and **ACDDoNotDisturb** properties indicate whether or not the extension is in a Do Not Disturb state for regular or ACD calls.
- The **Extension**, **FirstName**, and **LastName** properties identify the name and extension number for the user on this extension.

The **Refresh** method updates an Extension object's information.

Using the Device Status components

To use the Device Status components:

1. From Visual Basic add the "TeleVantage 3.5 Device Status Library" as a reference for the project. The Device Status library is contained in `TvStatus.dll`.
2. Create an instance of the Server component. Alternatively, Visual Basic's `CreateObject` function could be used to create the Server object.
3. Once the Server object is created, it needs to be initialized by calling the `Connect` method with the name of the Server to connect to. If the server machine is unavailable or the TeleVantage Server is not started, the `Connect` method will fail and raise a trappable error.

At this point everything is initialized and ready to be used. The Server object's `Devices` collection contains a Device object for each device on the system. Each Device object has a `AssignedExtensions` collection containing an Extension object for each extension assigned to the device.

The device status information (hook state, status, and currently logged in user) will be updated automatically as changes happen, and the `DeviceStateChanged` event will be raised immediately. Extension information (which comes from the Server database) will only be updated when the device `Refresh` or extension `Refresh` methods are called.

4. To terminate the application, call the `Disconnect` method of the Server object, which will release all existing Device Status objects. The `Disconnect` method will be called automatically if the Server object goes out of scope while it is connected.

See "The DeviceStatusSample Project" for an example of how these objects are called in a real program.

Required runtime files

The Device Status API has a number of dependencies that must be satisfied in order for it to run correctly:

Microsoft Visual Basic 6.0 Runtime:

| | |
|--------------|--------------|
| msvbvm60.dll | asycfilt.dll |
| oleaut32.dll | stdole2.tlb |
| olepro32.dll | comcat.dll |

Microsoft Active Data Objects (ADO):

Run the MDAC installer

TeleVantage Client/Server Communications:

| |
|-----------------|
| TeleVantage.tlb |
| TVSecBrg.exe |
| TVSecPS.dll |

The easiest way to get all of these components is to install the TeleVantage Administrator, Client, or TAPI Service Provider on the machine in question.

Device Status API quick reference

This section provides a brief description of each method in the Device Status API. Page numbers refer to the full “Device Status API Reference” in Appendix C. The full reference is also available in the TeleVantage SDK online help.

Server object

| | | |
|-----|--------------------------|---|
| C-2 | Connect method | Connects to the specified server and initializes the Server object. |
| C-2 | Disconnect method | Disconnects from the server. |
| C-2 | Devices property | Read only. Returns a collection of Device objects representing all devices on the current server. |
| C-3 | DeviceStateChanged event | Raised whenever the state of the device changes. |

Device object

| | | |
|-----|-----------------------------|---|
| C-3 | Refresh method | Refreshes information about the extensions on the device. |
| C-3 | AssignedExtensions property | Read only. Returns a collection of Extension objects representing all extensions assigned to this device. |
| C-4 | Class property | Read only. Returns the device's class (station or trunk). |
| C-4 | CurrentExtension property | Read only. Returns the Extension object currently logged in to the device. |
| C-4 | DefaultExtension property | Read only. Returns the Extension object that is the default for the device. |
| C-5 | HookState property | Read only. Returns the hook state of the device. |
| C-5 | Name property | Read only. Returns the name of the device (blank for stations and the trunk name for trunks). |
| C-5 | Number property | Read only. Returns the number of the device (positive for stations and negative for trunks). |
| C-6 | Status property | Read only. Returns a status code indicating the device's current activity. |

Extension object

| | | |
|-----|--------------------------|--|
| C-7 | Refresh method | Refreshes information about this extension. |
| C-7 | ACDDoNotDisturb property | Read only. Returns a boolean value indicating whether or not the extension is in ACD Do Not Disturb state. |
| C-7 | DoNotDisturb property | Read only. Returns a boolean value indicating whether or not the extension is in Do Not Disturb state. |
| C-7 | Extension property | Read only. Returns the extension number for this user. |
| C-8 | FirstName property | Read only. Returns the first name of the user on this extension. |
| C-8 | LastName property | Read only. Returns the last name of the user on this extension. |

CHAPTER 5

USING IN-BAND SIGNALING

CHAPTER CONTENTS

| | |
|--|-----|
| About in-band signaling with TeleVantage | 6-2 |
| TeleVantage telephone commands | 6-3 |

About in-band signaling with TeleVantage

In some cases, the TeleVantage SDK may not be the simplest way to implement extended functionality. If you just need a simple way to make calls or perform call control on a PC networked to the TeleVantage Server, and your application does not need to access digits or perform voice processing, you can use in-band signaling.

A human caller can send commands to any PBX, such as TeleVantage, by generating a flash (quickly pressing and releasing the hang up button on a telephone handset cradle). This signals the PBX that special instructions will follow, such as placing a call on hold or transferring a call to another extension. For example, a TeleVantage user would generate a flash and then press 1 to transfer a call.

This method of sending instructions is called “in-band signaling” because it travels over the same channel that voice is transmitted, in-band. Many telephony applications use this same method, communicating with PBX or Centrex systems through flash hook commands.

Most PBXs use their own proprietary sets of flash hook commands. An application that uses in-band signaling will usually provide you with a way to configure it for a specific set of flash hook commands, such as those used by TeleVantage. For example, **&3** (a flash followed by a 3) is the flash hook command that TeleVantage uses to disconnect from a call, so you would configure the in-band signaling application to use **&3** as its disconnect command. Other commands are listed below in “TeleVantage telephone commands” on page 6-3.

IVR applications connected this way still have access to Caller ID and DID, if the station is configured appropriately. Both options are set in the Administrator's Users view, under the Phone tab.

This approach requires the IVR application to use dedicated voice board resources that can either be in the TeleVantage Server or in a separate server. If the IVR application will be on the same server, the voice board resources must be dedicated to the IVR application, unlike IVR Plug-ins where voice resources are dynamically shared. Regardless of whether the IVR application runs on the TeleVantage Server or a separate server, the voice board's ports must be connected to one or more TeleVantage stations. This requires a TeleVantage station license and MSI hardware port to support a phone line for each TeleVantage station that is to be connected to the IVR application's voice board's ports.

To dedicate the voice board resources to your IVR application's needs, the Dialogic devices must be reserved (see “Reserving Dialogic Devices” in *Administering TeleVantage*). This allows an application that is running on the same server to obtain full control over these devices.

TeleVantage telephone commands

The commands listed below can be used to send in-band signaling commands to the TeleVantage Server. It is useful to try these commands manually before attempting to code them. See chapters 3 through 6 in *Using TeleVantage* for details.

Note: From within any Telephone Commands menu, you can press * to return to the higher level menu.

Call screening menu

If you have verbal call screening turned on, you are offered the following options when you answer an incoming call, either by picking up the phone or pressing **Flash** after hearing the call waiting beep.

| Call screening menu | |
|---------------------|--|
| 1 | Connect to the caller. |
| 2 | Send the caller to your voice mail. |
| 3 | Send the caller to voice mail and monitor the message being left. Press Flash 1 to pick up the call in mid-message. |
| 4 | In call waiting situations, create a conference call with your current call and the new one. |
| Hang up | Send the caller to the next step on your routing list. Unless you have changed your routing list, the next step is your voice mail, so hanging up is the same as pressing 2. |

Call handling menu

As soon as you press **Flash** to put a call on hold, you will hear a context-sensitive menu of call handling options. The caller does not hear these menu prompts. At any time while listening to the prompts, you can press **Flash** again to reconnect to the caller.

| Call Handling Commands With a call on the line, press Flash, then... | |
|---|--|
| 1 | Transfer the call |
| 2 | Send the call to voice mail |
| 3 | Disconnect from the call |
| 4 | Reconnect to the call |
| 5 | Create a conference call |
| 6 | Park the call |
| 7 | Silent hold (stops menu until next keypress) |
| 8 | Send a Flash to Centrex/PBX service beyond TeleVantage |
| # | Get a dial tone (start another call) |
| Flash | Reconnect to the call |

Quick call menu

When you press * from a dial tone, you are offered a menu of quick call commands for placing and answering calls. You can also press * plus the command directly:

| Quick call commands | |
|---------------------|--|
| *0 | Hear your phone's station ID and extension. |
| *10 | Enable and disable hands-free answering. |
| *11 | Enter an account code for the current call or the call you are about to dial. |
| *55 | Hear real-time call center statistics for the queue of your choice. |
| *66 | Redial the last call you placed. |
| *69 | Dial the phone number of your last incoming call. |
| *70 | Disable call waiting for the next call. |
| *91 | Answer another ringing phone. Enter the extension of the phone to answer. |
| *92 | Retrieve a parked call. |
| *93 | Dial a TeleVantage user by name. |
| *95 | Manage your calls on hold. |
| *96 | Log off from remote session. Only available at a dial tone when logged in from a remote phone. |
| *99 | Answer another ringing phone within your workgroup. See *91 also. |

Voice mail/Account menu commands

The Voice Mail/Account menu lets you listen to your voice mail, send messages to other users' voice mail, and change your TeleVantage account settings. You need to log in to access this menu.

To log in to a Voice Mail/Account menu, enter:

<extension number> # <password>

If the voice mailbox is almost full when you log in, you are given the option to empty your Deleted folder.

| Summary of Voice Mail/Account Menu Commands | |
|--|---|
| 1 | Voice messages (Inbox folder) |
| 2 | Voice messages (Saved folder) <ul style="list-style-type: none"> 1 Replay 2 Next message 3 Delete/Undelete message 41 Reply 42 Forward 43 Call back 5 Previous message 6 Save 7 Rewind 8 Undelete all 9 Fast Forward # Skip message preamble; skip rest of message |
| 3 | Send voice message <ul style="list-style-type: none"> 1 Send 2 Review 3 Re-record 4 Append 5 Mark Urgent 6 Mark Private * Cancel |
| 4 | Manage greetings <ul style="list-style-type: none"> 1 Replay 2 Next greeting 3 Make active 4 Re-record 5 Revert 6 Record new greeting 7 Delete |
| 5 | Call forwarding <ul style="list-style-type: none"> 1 To this ext. 2 To internal 3 To external 4 Cancel forwarding 5 Query forwarding 6 Toggle standard call rules |

Summary of Voice Mail/Account Menu Commands

| | |
|----------|---|
| 6 | Account preferences 1 Do Not Disturb 2 Record voice title 3 Change password |
| 7 | Have TeleVantage hang up |
| # | Dial tone to start another call |

Note: Option 7 is only available when logging in from an outside line.

Quick commands for call center agents

The following commands are available only for agents in a call center queue.

Quick call commands for call center agents

| | |
|------------|---|
| *14 | Mark the identity of all subsequent outbound calls during your shift, to keep track of which outbound calls are queue-related. To mark all your subsequent outbound calls as being from the queue, press *14<queue's extension># . To mark all your subsequent outbound calls as being from yourself, press *14# . |
| *51 | Sign in. The call center queues begin sending you calls. This command changes your personal status to Available (Queue Only). When you sign in with *51, only queue calls ring your phone. Non-queue calls are sent directly to your voice mail. To have all your calls ring your phone, sign in by choosing the personal status Available. Note: To receive queue calls, you must be on call with a queue and you must be signed in. Your TeleVantage system administrator can tell you the queues for which you are on call. |
| *52 | Sign out. The call center queues stop sending you calls. This command changes your personal status to Available (Non-Queue). After you sign out, non-queue calls continue to ring your phone. To prevent all calls from ringing your phone, use the Do Not Disturb personal status. |
| *53 | Go on break. The call center queues stop sending you calls, but you are still considered to be signed in for statistical purposes. This command changes your personal status to On Break. While you are on break, non-queue calls continue to ring your phone. |
| *54 | End wrap-up. This command terminates the wrap-up time that follows a queue call. During wrap-up time the queue does not you send calls. Ending wrap-up makes you available to receive queue calls again. |
| *55 | Hear real-time call center statistics for a queue. To use *55, you must have been given permission to do so by your TeleVantage system administrator. |

CHAPTER 6

THE TELEVANTAGE TAPI SERVICE PROVIDER

CHAPTER CONTENTS

| | |
|--|-----|
| About the TeleVantage TAPI Service Provider | 5-2 |
| TeleVantage TAPI Service Provider capabilities | 5-2 |
| TAPI and TeleVantage custom call data | 5-3 |

About the TeleVantage TAPI Service Provider

TAPI support makes it possible to integrate an existing TAPI application with the TeleVantage Server, and to use TAPI to make calls or perform call control on a PC networked to the TeleVantage Server.

You can use the TeleVantage TAPI Service Provider (TSP) to integrate your custom TAPI applications with the TeleVantage Server. Unlike IVR Plug-ins, which must run on the Server, TAPI applications run on any PC that has the TeleVantage TSP installed and is networked to the Server.

You install the TSP by running the Setup program in the Netsetup folder on the TeleVantage Server. (See *Installing TeleVantage* for details.) As part of the installation, the TeleVantage TAPI Configuration wizard prompts you to specify which TeleVantage station the TSP will monitor for incoming calls. After the TSP is installed, you can change the station at any time by choosing **Start > Programs > Artissoft TeleVantage > TAPI Configuration**. If you need to work with the station ID programmatically, refer to the description of `.\Client\TSP\Logon` in the configuration settings appendix of *Administering TeleVantage*.

Whenever a call is transferred to the station being monitored by the TeleVantage TSP, the TSP notifies the TAPI-compatible applications of the call. Once your TAPI application gets the call, it can get Caller ID or DID, transfer the call, put it on hold, park it, or hang up. TAPI applications can also place new calls, unpark calls, and set or get TeleVantage custom data. See "TAPI and TeleVantage custom call data" on page 5-3 for more information.

TeleVantage TSP can be used to build custom screen pop, dialing and call control applications. For example, you could pop up a window whenever an important caller rings. Or you could use the TSP to place calls from a database of numbers.

TeleVantage TAPI Service Provider capabilities

This section is intended for experienced TAPI developers who want to integrate their TAPI applications with TeleVantage. It assumes knowledge of Microsoft TAPI.

The TeleVantage TAPI Service Provider (TSP) is a TAPI 2.1-compliant TSP that provides basic and supplementary call control services on any Windows 98, ME, NT, 2000, or XP computer. The TeleVantage Client does not need to be installed on the computer.

The TSP does not expose programmable media services, which means that your TAPI application can not (among other things) perform voice processing or collect digits from the caller. If your application needs media services, we recommend that you use the IVR Plug-in interface.

The TSP requires, as a minimum, a running TeleVantage Server and a single TeleVantage MSI station. With this basic setup it can provide most of its call control functionality. The TSP exposes a single line which represents the configured TeleVantage station. It translates the various TAPI first person-call control requests (primarily `TSPI_lineMakeCall`) into a format that the TeleVantage Server understands. To the TeleVantage Server, the TSP appears as a normal TeleVantage Client. Call handling is therefore subject to the same dialing restrictions as the configured user.

Supported TAPI functions

The TSP supports the functions MakeCall, Dial, Drop, Hold, Unhold, SwapHold, Park, Supervised Transfer, and Blind Transfer. The TSP supports MakeCall using basic dial strings in standard TAPI dialable address format. It does not currently support IP addresses or any `LINECALLPARAMS` so if you need to call an IP address, call an extension that is routed to an IP address instead.

In order for outbound dialing on trunks to work properly, the appropriate TAPI dialing prefixes must be configured for the current Windows Telephony location. Each prefix must correspond to an access code for a TeleVantage Dialing Service or the call attempt will fail. The TSP Configuration Wizard attempts to validate these prefixes against the server and warns of possible problems in this area.

Supported bearer modes

The TSP reports support for LINEBEARERMODE_INTERACTIVEVOICE and LINEBEARERMODE_DATAMODEM. Artisoft recommends that TAPI applications that open the TeleVantage line use INTERACTIVEVOICE.

Call states

The TSP may report any of the following TAPI call states:

| | | |
|----------|-----------|--------------------|
| IDLE | RINGBACK | ONHOLDPENDTRANSFER |
| OFFERING | CONNECTED | PROCEEDING |
| ACCEPTED | BUSY | DISCONNECTED |
| DIALTONE | ONHOLD | UNKNOWN |
| DIALING | | |

Exported TSP functions

The following list contains all TSPI_xxx functions exposed by the TeleVantage TSP to TAPI:

| | |
|------------------------------------|-----------------------------------|
| TSPI_lineAnswer | TSPI_linePark |
| TSPI_lineBlindTransfer | TSPI_lineSetCallData |
| TSPI_lineClose | TSPI_lineSetDefaultMediaDetection |
| TSPI_lineCloseCall | TSPI_lineSetStatusMessages |
| TSPI_lineCompleteTransfer | TSPI_lineSetupTransfer |
| TSPI_lineConditionalMediaDetection | TSPI_lineSwapHold |
| TSPI_lineDial | TSPI_lineUnhold |
| TSPI_lineDrop | TSPI_lineUnpark |
| TSPI_lineGetAddressCaps | TSPI_providerConfig |
| TSPI_lineGetAddressStatus | TSPI_providerInit |
| TSPI_lineGetCallAddressID | TSPI_providerInstall |
| TSPI_lineGetCallInfo | TSPI_providerRemove |
| TSPI_lineGetCallStatus | TSPI_providerShutdown |
| TSPI_lineGetDevCaps | TSPI_providerEnumDevices |
| TSPI_lineGetID | TSPI_providerGenericDialogData |
| TSPI_lineGetLineDevStatus | TSPI_providerUIIdentify |
| TSPI_lineGetNumAddressIDs | |
| TSPI_lineHold | TUISPI_lineConfigDialog |
| TSPI_lineMakeCall | TUISPI_providerConfig |
| TSPI_lineNegotiateTSPIVersion | TUISPI_providerInstall |
| TSPI_lineOpen | TUISPI_providerRemove |

TAPI and TeleVantage custom call data

TeleVantage provides two COM interfaces to access TeleVantage custom call data: ITvTapiCustomData and ITvCallProperties.

Note: Even though the TSP implements TSPI_lineGetCallInfo and TSPI_lineSetCallData, you should not use these functions to access TeleVantage custom call data because they are used internally by the TSP. The ITvTapiCustomData and ITvCallProperties interfaces that you should use are part of the TSP itself. You do not need to install any additional components to support custom data through TAPI.

For example, in a Visual Basic TAPI application you would create a reference to the TeleVantage TAPI Type Library C:\WINNT\SYSTEM32\TVCLIENT.TSP. Once you have a valid TAPI call handle, you can put and get custom data as needed. Just like the IVR Plug-in custom data interface, this data is exposed to other TAPI clients if you transfer the call, other IVR Plug-ins and also the Client's Call Monitor if configured as described in "IVR Plug-ins and call data".

Appendixes

IVR PLUG-IN API REFERENCE

CHAPTER CONTENTS

| | |
|-----------------------------------|------|
| Overview | A-2 |
| PluginCaller interface | A-2 |
| PluginApplication interface | A-14 |
| IPluginServer interface | A-15 |
| PluginMedia interface | A-16 |

Overview

This appendix describes the interfaces, methods, properties, and events that make up the TeleVantage IVR Plug-in API. For more information about using the IVR Plug-in API, see Chapter 3.

The IVR Plug-in API consists of the following interfaces:

- **PluginCaller interface.** See page A-2.
- **PluginApplication interface.** See page A-14.
- **IPluginServer interface.** See page A-15.
- **PluginMedia interface.** See page A-16.

Note: For backward compatibility, previous versions of the IVR Plug-in API are supported, and existing IVR Plug-in applications will continue to run. See Appendix B for information on how to migrate existing IVR Plug-in applications to use the current IVR Plug-in API.

PluginCaller interface

The PluginCaller interface consists of the following methods and properties:

AssociateCalledExtension method (PluginCaller interface)

Description Attempts to associate the called ID with a system target based on extension.

Syntax `[Associated] = Object.AssociateCalledExtension Extension`

| Arguments | Name | Data Type | Description |
|----------------|------------|-----------|---|
| | Extension | String | The extension number of any system target that exists in the TeleVantage Server. |
| Returns | Associated | String | Returns True if the extension specified was a valid system target and the association completed successfully. |

Remarks When a call is created, the Server records the system target (User, Auto Attendant, Queue, IVR Plug-in, etc.) that was initially called. This method allows an IVR Plug-in to change that information to a different system target.

Example

AssociateCalledCallerID method (PluginCaller interface)

Description Attempts to associate the called ID with a personal or public contact.

Syntax `[Associated] = Object.AssociateCalledID CallerIdNumber, CallerIdName, ExtensionContext`

| Arguments | Name | Data Type | Description |
|-----------|----------------|-----------|--|
| | CallerIdNumber | String | Optional. The phone number of a TeleVantage contact you want to associate with. If this argument is not specified you must specify the CallerIDName argument. Default is "". |
| | CallerIdName | String | Optional. The name of a TeleVantage contact you want to associate with. If this argument is not specified you must specify the CallerIDNumber argument. Default is "". |

| | | | |
|----------------|--|---------|--|
| | ExtensionContext | String | Optional. Leave blank to match against TeleVantage public contacts or specify a user's extension number to match against the user's TeleVantage contacts. Default is "". |
| Returns | Associated | Boolean | Returns True if the method resulted in a matching contact and the association completed successfully. |
| Remarks | When a call is created to an external number, the Server searches the user's personal TeleVantage contacts, and if a match is not found, the Server's public contacts. This method allows an IVR Plug-in to change that information to a TeleVantage personal or public contact based on Caller ID name or number. | | |
| Example | | | |

AssociateCallerIDmethod (PluginCaller interface)

| | | | |
|--------------------|---|------------------|---|
| Description | Attempts to associate the Caller ID with a personal or public contact. | | |
| Syntax | [Associated] = Object.AssociateCallerID CallerIdNumber, CallerIdName, ExtensionContext | | |
| Arguments | Name | Data Type | Description |
| | CallerIdNumber | String | Optional. The Caller ID number that matches the Caller ID stored with the TeleVantage contact you want to associate with. If this argument is not specified you must specify the CallerIDName argument. Default is "". |
| | CallerIdName | String | Optional. The Caller ID name that matches the Caller ID name stored with the TeleVantage contact you want to associate with. If this argument is not specified you must specify the CallerIDNumber argument. Default is "". |
| | ExtensionContext | String | Optional. Leave blank to match against TeleVantage public contacts or specify a user's extension number to match against the user's TeleVantage contacts. Default is "". |
| Returns | Associated | Boolean | Returns True if the method resulted in a matching contact and the association completed successfully. |
| Remarks | This method associates a TeleVantage contact with the caller allowing IVR Plug-ins to perform their own contact identification. You can use this method to change calls that previously appeared as from Unknown to be from a TeleVantage personal or public contact. | | |
| Example | | | |

AssociateExtension method (PluginCaller interface)

Description Attempts to associate the Caller ID with a system target based on extension.

Syntax `Object.AssociateExtension`

| Arguments | Name | Data Type | Description |
|-----------|------------|-----------|---|
| | Extension | String | The extension number of any system target that exists in the TeleVantage Server. |
| | Associated | String | Returns True if the extension specified was a valid system target and the association completed successfully. |

Remarks This method associates a TeleVantage system target (User, Auto Attendant, Queue, IVR Plug-in, etc.) with the caller, effectively changing who the call appears to be from. You can use this method to change calls that previously appeared as from a user to be from a different user.

Example**CallDone method (PluginCaller interface)**

Description Returns control of the call to the Server with instructions on how it should continue processing the call.

Syntax `Object.CallDone Action, Number, Type, AccessCode, SubType`

| Arguments | Name | Data Type | Description |
|---------------------|------------|-----------------------|--|
| | Action | NextAction | Specifies the next step the TeleVantage Server should take to process the call. |
| Enumerations | | | |
| | | <i>Constant</i> | <i>Value</i> <i>Description</i> |
| | | naHangup | 0 Hang up call. |
| | | naTransferToNumber | 1 Transfer call to the extension or phone number specified in DialString, as if it was dialed from dialtone. |
| | | naTransferToVoiceMail | 2 Transfer call to voice mailbox of the extension specified in DialString. |
| | | naReturnToRoutingList | 3 Continue with next step in routing list. |
| | Number | String | Optional. Extension or phone number required if the Action argument is naTransferToNumber or naTransferToVoiceMail. The default is "". |
| | Type | AddressType | Optional. The type of address specified in the Number argument. The default is atExtension. Use atTelephone for phone numbers. See the section AddressType Enumerations for a complete list. |
| | AccessCode | String | Optional. Only used if the Type argument is atTelephone, atExtension (for PBX/Centrex dialing services) or for atInternet. Specify the Dialing Service access code (e.g. "9") to dial the number specified in the Number argument. The default is "", which will use the default phone number dialing service. |

| | | | |
|-----------------|--|------|--|
| | SubType | Long | Optional. Only used if the Type argument is <code>atTelephone</code> , specify <code>astUseServiceRules</code> if the number should be dialed using the Server's dialing rules or specify <code>astUnknown</code> to dial the number exactly as specified in the Number argument. The default is <code>astUnknown</code> . |
| Remarks | When this method finishes, the IVR Plug-in is no longer processing the call and any voice resources used are reclaimed by the Server. This is the last operation an IVR Plug-in can perform before it is terminated, so after calling this method, do not call any additional IVR Plug-in methods. | | |
| Examples | Transfer the caller to the voice mailbox at extension 102: <pre>mPlugInCaller.CallDone naTransferToVoiceMail, "102"</pre> Transfer the caller to the external number 6175551212: <pre>mPlugInCaller.CallDone naTransferToNumber, "6175551212", atTelephone, "", astUseServiceRules</pre> | | |

Delete method (PluginCaller interface)

| | | | |
|--------------------|---|------------------|--|
| Description | Deletes the specified custom data property from the party in the call. | | |
| Syntax | <code>Object.Delete PropertyName</code> | | |
| Arguments | Name | Data Type | Description |
| | PropertyName | String | The name of a custom data variable that you want to remove from the party in the call. |
| Remarks | If the party in the call has a custom data variable such as CustomerID, you can use this method to remove it. | | |
| Example | <pre>mPlugInCaller.Delete "CustomerID"</pre> | | |

DeviceName property (PluginCaller interface)

| | |
|--------------------|---|
| Description | Returns the name of the allocated raw voice resource, if any. |
| Syntax | <code>[DeviceName] = Object.DeviceName</code> |
| Remarks | This property is only filled in if using the <code>GetDevice</code> method and is not filled in if using the <code>GetMedia</code> method |
| Example | <pre>mPlugInCaller.SetStatus "The voice resource being used is "& mPlugInCaller.DeviceName</pre> |

Exists method (PluginCaller interface)

| | | | |
|--------------------|--|------------------|---|
| Description | Returns True if the specified custom data property exists on the party in the call. | | |
| Syntax | <code>[Exists] = Object.Exists PropertyName</code> | | |
| Arguments | Name | Data Type | Description |
| | PropertyName | String | The name of a custom data variable. |
| Returns | Exists | Boolean | Returns True if the specified custom data property exists on the party in the call. |
| Remarks | | | |
| Example | <pre>If mPlugInCaller.Exists "CustomerID" then mPlugInCaller.SetStatus "The caller has a CustomerID variable" End If</pre> | | |

Get method (PluginCaller interface)

| | | | |
|--------------------|---|---|--|
| Description | Gets a standard or custom party attribute from the party in the call, for example, CallerID or CustomerID | | |
| Syntax | <code>[Value] = Object.Get Keyname, DefaultValue</code> | | |
| Arguments | Name | Data Type | Description |
| | Keyname | String | The name of a custom data variable or the name of a standard party attribute from the table below. |
| | DefaultValue | String | Optional. If the Keyname specified does not exist on the party, the Value returned will be set to the DefaultValue argument. Default is "" |
| Returns | Value | String | Returns the value of the Keyname specified. |
| Remarks | Use this method to obtain the value of any standard or custom data item attached to the party in the call. To set a standard or custom data value see the Set method. The names of all custom data variables are defined in the Administrator. The names of all standard party attributes are listed below. | | |
| | Standard Property | Description | |
| | ppHandle | The handle of the party | |
| | ppDeviceNumber | The device number that the party is on. Positive numbers are stations, negative numbers are trunks. | |
| | ppLineAppearanceDevice | The device number which owns the line appearance. | |
| | ppLineAppearanceNumber | Line appearance number | |
| | ppDirection | The direction of the party, either inbound or outbound. | |
| | ppStartTime | Time the party entered the current call | |
| | ppQueueWaitTime | Total time in seconds that party has waited in call center queues. | |
| | ppID | The ID of the party that can be passed to the client API. Use one of the AssociateXXX methods to change this value. | |
| | ppExtension | The extension of the party. Use one of the AssociateXXX methods to change this value | |

| | |
|----------------------------|---|
| ppLastName | The party's last name. |
| ppFirstName | The party's first name. |
| ppDisplayName | Suggested display name for the party, a combination of FirstName and LastName. |
| ppVoiceTitleFile | Server local path and file name of the party's recorded voice title. A non-empty path does not guarantee that the file actually exists. |
| ppNotes | Notes about the party. |
| ppAccountCode | The account code associated with the party. Account code validation is ignored when writing to this property. |
| ppCallerIDName | Caller ID name supplied by the party owner or the telephone company. |
| ppCallerIDNumberType | The type of the party's address. See the AddressType enumeration for possible values. |
| ppCallerIDNumber | The party's address. This is the Caller ID number for inbound telephone calls. CallerIDNumberType indicates if it will be a phone number, extension, or IP address. |
| ppCallerIDNumberAccessCode | The access code of the Dialing Service used by the party's address, if applicable. |
| ppCallerIDNumberSubType | Only used if xxxNumberType is atTelephone. astUseServiceRules indicates the number was dialed using the Server's dialing rules. astUnknown indicates the number was dialed exactly as specified in xxxNumber. |
| ppCallbackNumberType | The type of the party's callback address. See the AddressType enumeration for possible values. |
| ppCallbackNumber | The party's callback address. This is the Caller ID number for inbound telephone calls. CallerIDNumberType indicates if it will be a phone number, extension, or IP address. |
| ppCallbackNumberAccessCode | The access code of the Dialing Service used by the party's callback address, if applicable. |
| ppCallbackNumberSubType | Only used if xxxNumberType is atTelephone. astUseServiceRules indicates the number was dialed using the Server's dialing rules. astUnknown indicates the number was dialed exactly as specified in xxxNumber. |
| ppCalledID | The ID of the called party that can be passed to the client API. Use one of the AssociateXXX methods to change this value. |
| ppCalledExtension | The extension of the called party. Use one of the AssociateXXX methods to change this value |
| ppCalledLastName | The called party's first name. |
| ppCalledFirstName | The called party's last name. |
| ppCalledDisplayName | Suggested display name for the called party, a combination of FirstName and LastName. |
| ppCalledNumberType | The type of the called party's address. See the AddressType enumeration for possible values. |
| ppCalledNumber | The called party's address. This is the Caller ID number for inbound telephone calls. CallerIDNumberType indicates if it will be a phone number, extension, or IP address. |

| | |
|--------------------------|---|
| ppCalledNumberAccessCode | The access code of the Dialing Service used by the called party's address, if applicable. |
| ppCalledNumberSubType | Only used if xxxNumberType is atTelephone. astUseServiceRules indicates the number was dialed using the Server's dialing rules. astUnknown indicates the number was dialed exactly as specified in xxxNumber. |
| ppHandlerID | The ID of the party whom handled the call (placed by or answered by) that can be passed to the client API. Use one of the AssociateXXX methods to change this value. |
| ppHandlerExtension | The extension of the party whom handled the call (placed by or answered by). Use one of the AssociateXXX methods to change this value |
| ppHandlerLastName | The first name of the party whom handled the call (placed by or answered by). |
| ppHandlerFirstName | The last name of the party whom handled the call (placed by or answered by). |
| ppHandlerDisplayName | Suggested display name for the of the party whom handled the call (placed by or answered by), a combination of FirstName and LastName. |

Examples

```
' Get a standard party property from the caller
Dim CallerIDName as String
CallerIDName = mPlugInCaller.Get ppCallerIDName
mPlugInCaller.SetStatus "The Caller's Caller ID Name is " & CallerIDName

' Get a custom data variable that may have been set by an auto attendant, queue
' or another plug-in. If the customer ID is not set on the caller, use "N/A" as a default
Dim CallersCustomerID as String
CallersCustomerID = mPlugInCaller.Get "CustomerID", "N/A"
mPlugInCaller.SetStatus "The Caller's customer ID is " & CallersCustomerID
```

GetDevice method (PluginCaller interface)

| | | | |
|--------------------|---|------------------|--|
| Description | Allocates a raw voice resource from the TeleVantage Server's device pool so that you can pass it to a third party voice processing toolkit. | | |
| Syntax | [DeviceName] = Object. GetDevice ExtraInfo | | |
| Arguments | Name | Data Type | Description |
| | ExtraInfo | String | Optional. Reserved for future use. |
| Returns | DeviceName | String | Returns the raw voice resource device name for advanced voice processing by a 3rd party telephony toolkit. |
| Remarks | Use this method if you need to use a 3rd party telephony toolkit such as Intel's CallSuite or the low level Dialogic C API to perform voice processing commands that exceed the capabilities of the IVR Plug-in API's media object. 3rd party telephony toolkits will require the name of the raw voice processing device in order to perform any media processing. | | |
| | It is possible to use both a media object and a 3rd party telephony toolkit in the same IVR Plug-in, but not at the same time. If you previously called the GetMedia method to obtain a media object you must first destroy the Media object before calling this method and using the 3rd party telephony toolkit. Likewise after calling the GetDevice method, you must call the ReleaseDevice method before calling the GetMedia method to use the media object's voice processing methods. | | |
| Example | <pre>Dim DeviceName as String DeviceName = mPlugInCaller.GetDevice</pre> | | |

GetMedia method (PluginCaller interface)

Description Allocates a voice resource and returns a PluginMedia object associated with it.

Syntax [Media] = Object.GetMedia ExtraInfo

| Arguments | Name | Data Type | Description |
|------------------|-------------|------------------|------------------------------------|
| | ExtraInfo | String | Optional. Reserved for future use. |

Returns Media PluginMedia Returns a PlugInMedia object for basic voice processing.

Remarks With a PlugInMedia object you can perform basic voice processing tasks such as play and record files, get digits, play tones, and speak numbers, dates, times, and more. See the PlugInMedia interface for more information.

See the GetDevice method if you need to perform advanced voice processing tasks with a 3rd party telephony toolkit.

Example

```
Dim mTVMedia as PluginMedia
mTVMedia = mPlugInCaller.GetMedia
mTVMedia.PlayFile "Hello.vox"
```

GetXmitTimeslot method (PluginCaller interface)

Description Returns the TDM bus transmit timeslot of the caller's device.

Syntax Timeslot = Object.GetXmitTimeslot

| Arguments | Name | Data Type | Description |
|------------------|-------------|------------------|---|
| | Timeslot | Long | Returns the TDM bus transmit timeslot of the caller's device. |

Remarks Use this method if you need access to the TDM bus transmit timeslot of the caller's device so you could later route another TDM device to listen to the caller's transmitting audio. For example, have a fax device listen to the transmitting audio of a fax call.

Example

Listen method (PluginCaller interface)

Description Performs a half duplex route of the media stream, such that the caller's device listens to the specified TDM bus timeslot.

Syntax Object.Listen Timeslot

| Arguments | Name | Data Type | Description |
|------------------|-------------|------------------|--|
| | Timeslot | Long | The TDM bus timeslot number you want the calling party to listen to. |

Remarks

Example

Ping method (PluginCaller interface)

| | |
|--------------------|---|
| Description | Calling this method tells the server the IVR Plug-in is still running properly. |
| Syntax | <code>Object.Ping</code> |
| Remarks | Calling any method on PluginCaller interface executes an implicit Ping. The Server will terminate IVR Plug-in's which fail to call this method or any PluginCaller method for a 4 minutes. The 4 minute time period can be changed via a setting on the Server. |
| Example | <pre>Do 'perform some time consuming processing, such as a database query 'Ping the Server so it doesn't terminate the Plug-in while this processing is happening. mPluginCaller.Ping Until TheProcessingIsDone</pre> |

PluginComments property (PluginCaller interface)

| | |
|--------------------|---|
| Description | The comments for the called IVR Plug-in as defined in the Administrator. |
| Syntax | <code>[DID] = Object.PluginComments</code> |
| Remarks | This method allows you to retrieve the comments specified for the current IVR Plug-in as defined in the Administrator's IVR Plug-in view. |
| Example | <pre>Dim MyPlugInsComments as String MyPlugInsComments = mPluginCaller.PluginComments</pre> |

PluginDID property (PluginCaller interface)

| | |
|--------------------|---|
| Description | The DID number of the called IVR Plug-in as defined in the Administrator. |
| Syntax | <code>[DID] = Object.PluginDID</code> |
| Remarks | This method allows you to retrieve the DID number specified for the current IVR Plug-in as defined in the Administrator's IVR Plug-in view. |
| Example | <pre>Dim MyPlugInsDIDNumber as String MyPlugInsDIDNumber = mPluginCaller.PluginDID</pre> |

PluginExtension property (PluginCaller interface)

| | |
|--------------------|---|
| Description | The extension of the called IVR Plug-in as defined in the Administrator. |
| Syntax | <code>[Extension] = Object.PluginExtension</code> |
| Remarks | This method allows you to retrieve the extension number specified for the current IVR Plug-in as defined in the Administrator's IVR Plug-in view. |
| Example | <pre>Dim MyPlugInsExtension as String MyPlugInsExtension = mPluginCaller.PluginExtension</pre> |

PluginName property (PluginCaller interface)

- Description** The Name of the called IVR Plug-in as defined in the Administrator.
- Syntax** [Name] = Object.PluginName
- Remarks** This method allows you to retrieve the Name specified for the current IVR Plug-in as defined in the Administrator's IVR Plug-in view.
- Example**
Dim MyPlugInsName as String
MyPlugInsName = mPluginCaller.PluginName

PluginVariable property (PluginCaller interface)

- Description** The custom data variable name of the called IVR Plug-in as defined in the Administrator.
- Syntax** [VarName] = Object.PluginVariable
- Remarks** This method allows you to retrieve the custom data variable name specified for the current IVR Plug-in as defined in the Administrator's IVR Plug-in view.
- Example**
Dim MyPlugInsVariable as String
MyPlugInsVariable = mPluginCaller.PluginVariable

ReleaseDevice method (PluginCaller interface)

- Description** Returns a raw voice resource to the Server's device pool.
- Syntax** Object.ReleaseDevice DeviceName
- | Arguments | Name | Data Type | Description |
|-----------|------------|-----------|--|
| | DeviceName | String | Device name voice resource to release. |
- Remarks** See the GetDevice method
- Example**

Set method (PluginCaller interface)

Description Sets a custom or standard party attribute for the party in the call, for example, CustomerID or AccountCode.

Syntax `Object.set KeyName, Value`

| Arguments | Name | Data Type | Description |
|-----------|---------|-----------|--|
| | Keyname | String | The name of a custom data variable or the name of a standard party attribute from the table below. |
| | Value | String | The new value of the Keyname specified. |

Remarks Use this method to set the value of many standard or any custom data item attached to the party in the call. To get a standard or custom data value see the get method. The names of all custom data variables are defined in the Administrator. The names of all writable standard party attributes are listed below.

| Standard Property | Description |
|----------------------------|---|
| ppNotes | Notes about the party. |
| ppAccountCode | The account code associated with the party. Account code validation is ignored when writing to this property. |
| ppCallerIDName | Caller ID name supplied by the party owner or the telephone company. |
| ppCallerIDNumber | The party's address. This is the Caller ID number for inbound telephone calls. CallerIDNumberType indicates if it will be a phone number, extension, or IP address. |
| ppCallerIDNumberAccessCode | The access code of the Dialing Service used by the party's address, if applicable. |
| ppCallerIDNumberSubType | Only used if xxxNumberType is atTelephone. astUseServiceRules indicates the number was dialed using the Server's dialing rules. astUnknown indicates the number was dialed exactly as specified in xxxNumber. |
| ppCallbackNumberType | The type of the party's callback address. See the AddressType enumeration for possible values. |
| ppCallbackNumber | The party's callback address. This is the Caller ID number for inbound telephone calls. CallerIDNumberType indicates if it will be a phone number, extension, or IP address. |
| ppCallbackNumberAccessCode | The access code of the Dialing Service used by the party's callback address, if applicable. |
| ppCallbackNumberSubType | Only used if xxxNumberType is atTelephone. astUseServiceRules indicates the number was dialed using the Server's dialing rules. astUnknown indicates the number was dialed exactly as specified in xxxNumber. |
| ppCalledNumberType | The type of the called party's address. See the AddressType enumeration for possible values. |
| ppCalledNumber | The called party's address. This is the Caller ID number for inbound telephone calls. CallerIDNumberType indicates if it will be a phone number, extension, or IP address. |
| ppCalledNumberAccessCode | The access code of the Dialing Service used by the called party's address, if applicable. |

ppCalledNumberSubType Only used if xxxNumberType is atTelephone. astUseServiceRules indicates the number was dialed using the Server's dialing rules. astUnknown indicates the number was dialed exactly as specified in xxxNumber.

Examples

```
' Set a standard party property
mPlugInCaller.Set ppCallerIDName, "John Doe"

' Set a custom data variable so it can be displayed in the call monitor
mPlugInCaller.Set "CustomerID", "3057392"
```

SetStatus method (PluginCaller interface)

Description Sets the status string displayed in the Device Monitor on the Server or in the Administrator.

Syntax Object.SetStatus Status

| Arguments | Name | Data Type | Description |
|-----------|--------|-----------|---|
| | Status | String | Text to be displayed in the Device Monitor's status column for the trunk or station currently connected to the IVR Plug-in. |

Remarks Using SetStatus, you can visually monitor a Plug-in's activity. The Device Monitor shows the current status of every trunk and station in the Server. When a caller is connected to an IVR Plug-in, calls to the SetStatus method change the status displayed for the trunk or station on which the IVR Plug-in is operating. With this information Administrators or developers can monitor its activity even though they are not on the call. Each call to SetStatus changes the current status to the new status.

Example

```
mPlugInCaller.SetStatus "Asking the caller for the customer ID number"
mPlugInMedia.PlayFile "Please enter your customer id.vox"
mPlugInCaller.SetStatus "Waiting for the caller to enter their customer ID"
CustomerID = mPlugInMedia.GetDigits 5
```

PluginApplication interface

The PluginApplication interface consists of the following method:

- CallOffering method
- CallPlaced method
- CallTerminated method
-

CallOffering method (PluginApplication interface)

Description Invoked when the call is offered to the Plug-in.

Declaration `Private Sub PluginApplication_CallOffering(Caller as PluginCaller)`

| Inputs | Argument | Description |
|---------------|-----------------|---|
| | Caller | A PlugInCaller object that contains the details of the party calling. See the PlugInCaller object reference for more information. |

Remarks Before CallOffering is invoked, TeleVantage creates an instance of the Plug-in's class via its program ID. The program ID is specified in the Administrator's IVR Plug-ins view.

CallPlaced method (PluginApplication interface)

Description Invoked by the Server to notify the Plug-in when an outbound call has been dialed.

Declaration `Private Sub PluginApplication_CallPlaced(Caller as PluginCaller, CPType CallProgress)`

| Inputs | Argument | Description |
|---------------|-----------------|---|
| | Caller | A PlugInCaller object that contains the details of the party calling. See the PlugInCaller object reference for more information. |
| | CallProgress | CPType value. See the CPType enumeration for more information. |

Remarks TeleVantage does not do any call progress analysis before invoking this method. The number has been dialed on the first available trunk specified by the access code in the PlaceCall method. It is the responsibility of the Plug-in to initiate call progress analysis if required using the GetCallProgress method or a 3rd party telephony toolkit. Refer to the ReceiveCall and PlaceCall Visual Basic sample projects in the OutBoundCall directory.

CallTerminated method (PluginApplication interface)

Description Invoked when TeleVantage wants to terminate the Plug-in, for example when TeleVantage detects that the caller has hung up.

Declaration `Private Sub PluginApplication_CallTerminated()`

Remarks This method is invoked when the caller hangs up or if the Plug-in becomes unresponsive for 4 minutes.

In this method you should make sure you stop any voice processing (e.g. if using the Media object you should call the Stop method) and perform any necessary clean up. After CallTerminated is invoked, TeleVantage removes its reference to the ActiveX EXE's class, which in Visual Basic will then invoke the Class_Terminate subroutine.

CallTerminated is not invoked after the IVR Plug-In CallDone method.

IPluginServer interface

The IPluginServer interface consists of the PlaceCall method.

PlaceCall method (IPluginServer interface)

Description Allows the IVR Plug-in to make an outbound call, send data, and return control back to the TeleVantage Server.

Syntax [TvDevId] = Object.PlaceCall AccessCode, DialString, Extension, Subtype, InitialData

| Arguments | Name | Data Type | Description |
|------------------|-------------|------------------|--|
| | AccessCode | String | Access code for the dialing service used to make the outbound call. |
| | DialString | String | Phone number the TeleVantage Server will dial. The format of the number dialed depends on the selected Subtype, described below. |
| | Extension | String | The extension number to transfer the call to after dialing is complete. If you want to perform voice processing on the call, use the extension number of an IVR Plug-in. |
| | Subtype | AddressSubtype | Address subtype for the outbound call. See the AddressSubtype enumeration for more information. |
| | InitialData | String | User-defined string to pass to the target if an IVR Plug-in's extension is specified in the Extension argument.. |

Returns TvDevId Long The TeleVantage Device ID that the call was placed on.

Remarks See the PlaceCall and ReceiveCall sample for detailed information on how to use this method.

Example The following code initiates an outbound call:

```
Dim devid As Long
Dim mServer As IPluginServer
Set mServer = New IVRPlugin
devid = mServer.PlaceCall("9", "16173540600", "333", 0, "MyInitialDataString")
```

PluginMedia interface

The PluginMedia interface consists of the following methods:

- **Dial method.** Plays DTMF or MF digits. See page A-16.
- **FlushDigitBuffer method.** Returns a string representing all digits currently in the voice resource's digit buffer, and leaves the digit buffer empty. See page A-17.
- **GetCallProgress method.** Performs inbound call progress analysis. See page A-17.
- **GetDigits method.** Returns a string of digits from the digit buffer representing a number of touchtone digits, or a series of digits ending with a terminating digit. See page A-18.
- **GetXmitTimeslot method.** Returns the TDM bus transmit timeslot of the voice resource. See page A-19.
- **Listen method.** Performs a half duplex route such that the voice resource listens to the specified TDM bus timeslot. See page A-19.
- **PlayFile method.** Plays a pre-recorded audio file. See page A-20.
- **PlayString method.** Plays a voice string. See page A-21.
- **PlayTone method.** Plays a pre-defined tone such as dial tone or a busy signal. See page A-23.
- **RecordFile method.** Records a voice file. See page A-24.
- **SetRate method.** Set audio rate for the next or current playback operation. See page A-25.
- **SetVolume method.** Set audio volume for the next or current playback operation. See page A-25.
- **Stop method.** Stops all voice activity. See page A-25.

All PluginMedia methods are synchronous. All timeouts and durations are in milliseconds.

Dial method (PluginMedia interface)

| | | | |
|--------------------|---|------------------|--------------------|
| Description | Plays DTMF or MF digits. | | |
| Syntax | Object.Dial DialString | | |
| Arguments | Name | Data Type | Description |
| | DialString | String | Digits to dial. |
| Remarks | Use the PlaceCall method to call numbers. Use this method to play DTMF digits to the caller. A common use is to play sequences of DTMF digits to another IVR application. | | |
| Example | mTVMedia.Dial "102#928472#*" | | |

FlushDigitBuffer method (PluginMedia interface)

Description Returns a string representing all digits currently in the voice resource's digit buffer, and leaves the digit buffer empty.

Syntax [DigitBuffer]= Object.FlushDigitBuffer

| Arguments | Name | Data Type | Description |
|-----------|------|-----------|-------------|
|-----------|------|-----------|-------------|

| | | | |
|----------------|-------------|--------|--|
| Returns | DigitBuffer | String | String of digits retrieved from the voice resource's digit buffer. |
|----------------|-------------|--------|--|

Remarks The voice resource's digit buffer collects and stores digits detected over the line. The **FlushDigitBuffer** method retrieves all the digits already in the digit buffer without waiting for any conditions. The digit buffer is empty after this method completes.

The **FlushDigitBuffer** method is similar to the **GetDigits** method. However, the **GetDigits** method waits for conditions before retrieving digits (see page A-18.)

Example The following code retrieves all the digits in the digit buffer and places them in the Choice variable:

```
Choice = mTVMedia.FlushDigitBuffer
```

GetCallProgress method (PluginMedia interface)

Description Performs call progress analysis.

Syntax [ActionResult] = Object.GetCallProgress CallTimeout, ConnectLength, ConnectType

| Argument | Data Type | Description |
|----------|-----------|-------------|
|----------|-----------|-------------|

| | | |
|-------------|------|---|
| CallTimeOut | Long | Optional. Milliseconds to wait for call progress detection to complete. Default is 20,000 (20 seconds). |
|-------------|------|---|

| | | |
|---------------|------|--|
| ConnectLength | Long | Option return value that if connected, provides the length of the greeting salutation in milliseconds (e.g. "Hello" is about 1000 milliseconds long). Using this parameter it is possible to determine if you have reached an answering machine or a person, since answering machine greetings are much longer than live people. |
|---------------|------|--|

| | | |
|-------------|----------------------|--|
| ConnectType | ConnectTypeConstants | Option return value that returns the type of connection detected, if a connection was made. See ConnectTypeConstants enumeration for more information. |
|-------------|----------------------|--|

| | | | |
|----------------|--------------|--------------|---|
| Returns | ActionResult | ActionResult | See the ActionResult enumeration for a complete list of possible results for this method. |
|----------------|--------------|--------------|---|

Remarks A CallTimeOut of 0 seconds is no timeout at all and the method will immediately return. A CallTimeOut of -1 is an indefinite timeout.

Example

```
If mTVMedia.GetCallProgress(30000) = arBusy then
    mPlugInCaller.CallDone naHangUp
Else
    mTVMedia.PlayFile "Hello.vox"
End If
```

GetDigits method (PluginMedia interface)

Description Returns a string of digits from the digit buffer representing a number of touchtone digits, or a series of digits ending with a terminating digit. This method waits indefinitely (by default 15 seconds) until one of the conditions in the input list is met.

Syntax [DigitsBuffer] = Object.**GetDigits** MaxDigits, TermDigits, DigitsTimeOut, InterDigitTimeOut, InterDigitStart, Result

| Arguments | Name | Data Type | Description |
|----------------|-------------------|--------------|---|
| | MaxDigits | Integer | Optional. Maximum number of digits to collect. Default is 1. |
| | TermDigits | String | Optional. String indicating any digits that will terminate the method. To automatically strip off the terminating digit, add a "-" to the end of the string. Default is "". |
| | DigitsTimeOut | Integer | Optional. Maximum number of milliseconds the method will wait before returning. Default is 15000 milliseconds (15 seconds). |
| | InterDigitTimeOut | Integer | Optional. Maximum number of milliseconds this method will wait between digits before returning. Default is 0, no timeout set. |
| | InterDigitStart | Integer | Optional. Set to 1 to delay the InterDigitTimeout until after the first digit is collected. Default is 0. |
| | Result | ActionResult | Optional returned value. See the ActionResult enumeration for more information. |
| Returns | DigitBuffer | String | Returns a string of digits retrieved from the voice resource's digit buffer. |

Remarks A voice resource has a digit buffer that collects and stores digits detected over the phone line, such as touchtone digits. The **GetDigits** method retrieves digits from the digit buffer according to certain conditions. The following conditions can cause the **GetDigits** method to terminate:

The number of digits received is equal to the MaxDigits argument.

One of the digits in the TermDigits argument is detected. You can add a "-" to the end of the TermDigits argument to remove the terminating digit.

The time elapsed exceeds the DigitsTimeOut argument.

The time elapsed since the last digit was detected exceeds the InterDigitTimeOut argument.

The **Stop** method is executed.

The caller hangs up.

The digits retrieved by this method are removed from the digit buffer after this method executes. Any extra digits remain in the digit buffer for later retrieval, for example if 5 digits are in the digit buffer, but the MaxDigits argument was set to 4.

A InterDigitTimeOut or DigitsTimeOut of 0 is no timeout at all and the method will immediately return. A timeout of -1 is an indefinite timeout.

The **GetDigits** method is similar to the **FlushDigitBuffer** method. However, the **FlushDigitBuffer** method does not wait for conditions before retrieving digits (see page A-17.)

Example 1 The following code retrieves 10 digits and places them into the Choice variable, unless the caller presses # to terminate input or waits 20 seconds. If the caller presses #, the digits up to and including the # are returned. If the caller waits 20 seconds, no digits are returned, even if 9 digits were pressed.

```
Choice = mTVMedia.GetDigits(10, "#", 20000)
```

Example 2 The following code plays a voice file that prompts callers for their PIN number, and then retrieves it.

```
'Please enter your PIN Code followed by the pound sign
mTVMedia.PlayFile "ENTRPIN.VOX"
'specifying a 0 for MaxDigits allows a variable number of 'digits to be entered
'# terminates digit retrieval'- strips off the terminating digit
CustomerPIN = mTVMedia.GetDigits(0, "#-")
```

Example 3 The following code illustrates how digits are collected and removed from the digit buffer, while the remaining digits stay in the buffer awaiting a subsequent **GetDigits** method.

```
'Caller presses 12345
nowChoice = mTVMedia.GetDigits(2)
'Choice is now "12"
"NextChoice = mTVMedia.GetDigits(2)
'NextChoice is now "34"
LastChoice = mTVMedia.GetDigits(2)
```

The line above waits for the second digit to be pressed because only 1 digit (a "5") is in the digit buffer when the code executes. It waits as long as the default specified in the `DigitsTimeOut` argument. If no digit is pressed within that period of time, `LastChoice` is set to "" and the "5" remains in the digit buffer. It could be retrieved by the next **GetDigits** method; however, if we assume that the caller presses 678 before the timeout, `LastChoice` is now "56".

```
ReallyLastChoice = mTVMedia.GetDigits(2)
```

`ReallyLastChoice` is now "78" and no digits remain in the buffer.

GetXmitTimeslot method (PluginMedia interface)

Description Returns the TDM bus transmit timeslot of the voice resource.

Syntax [Timeslot] = Object.**GetXmitTimeslot**

| Arguments | Name | Data Type | Description |
|------------------|-------------|------------------|--------------------|
|------------------|-------------|------------------|--------------------|

| | | | |
|----------------|----------|---------|--|
| Returns | Timeslot | Integer | Returns the TDM bus transmit timeslot of the voice resource. |
|----------------|----------|---------|--|

Remarks

Example

Listen method (PluginMedia interface)

Description Performs a half duplex route such that the voice resource listens to the specified TDM bus timeslot.

Syntax Object.**Listen** Timeslot Timeslot

| Arguments | Name | Data Type | Description |
|------------------|-------------|------------------|--------------------|
|------------------|-------------|------------------|--------------------|

| | | | |
|--|----------|---------|--------------------------------|
| | Timeslot | Integer | TDM bus timeslot to listen to. |
|--|----------|---------|--------------------------------|

Remarks

Example

PlayFile method (PluginMedia interface)

Description Plays a pre-recorded audio file.

Syntax [Result] = Object.**PlayFile** FileName, TermDigits StartPosition, EndPosition, ClearDigitBuffer, Format

| Arguments | Name | Data Type | Description |
|-----------|------------------|------------|--|
| | FileName | String | File name to be played. |
| | TermDigits | String | Optional. Terminating digit mask. Default is "@" – all digits will terminate playback. |
| | StartPosition | Integer | Optional. Position within the file in bytes to start playing. Default is 0. |
| | EndPosition | Integer | Optional returned value. Position within the file in bytes where playback stopped. |
| | ClearDigitBuffer | Boolean | Optional. If True, clears the digit buffer before playback. Default is -1 (True). |
| | Format | FileFormat | Optional. File format of the audio file. FileFormat can be any of the following. The default format is 3, or 8KHz PCM. |

Enumerations:

| Constant | Value | Description |
|----------------|-------|-----------------------|
| ffVOX_ADPCM_6K | 0 | 6KHz VOX ADPCM format |
| ffVOX_ADPCM_8K | 1 | 8KHz VOX ADPCM format |
| ffVOX_PCM_6K | 2 | 6KHz VOX PCM format |
| ffVOX_PCM_8K | 3 | 8KHz VOX PCM format |
| ffWAV_PCM_9K | 6 | 6KHz WAV PCM format |
| ffWAV_PCM_8K | 7 | 8KHz WAV PCM format |
| ffWAV_PCM_11K | 8 | 11KHz WAV PCM format |

All file formats are 1 channel (mono). See Remarks for more information about each format.

Returns Result ActionResult Returns information on how the method completed. See the ActionResult enumeration for more information

Remarks This method clears the digit buffer before playing if the ClearDigitBuffer argument is set to -1 (the default value). Play always begins at the start of the voice file unless the StartPosition argument contains a valid non-zero value.

Play can be interrupted by any of the following conditions:

A valid terminating digit specified in the TermDigits argument is pressed (or is already in the digit buffer if ClearDigitBuffer is set to 0, False).

The caller hangs up.

The **Stop** method is invoked.

The FileFormat argument specifies the format for the file to be played. The following table lists some additional metrics for each file format:

| FileFormat | Kbps | Minutes/MB | Bytes/Second |
|-----------------------|------|------------|--------------|
| 6KHz VOX ADPCM format | 24 | 5.83 | 3000 |

| | | | |
|-----------------------|----|------|-------|
| 8KHz VOX ADPCM format | 32 | 4.37 | 4000 |
| 6KHz VOX PCM format | 48 | 2.91 | 6000 |
| 8KHz VOX PCM format | 64 | 2.18 | 8000 |
| 6KHz WAV PCM format | 48 | 2.91 | 6000 |
| 8KHz WAV PCM format | 64 | 2.18 | 8000 |
| 11KHz WAV PCM format | 88 | 1.59 | 11000 |

PlayString method (PluginMedia interface)

| Description | Plays a voice string. A voice string is a collection of one or more voice files played smoothly together using various play methods. Voice files can be prompts, numbers, dates, times, characters or ordinal numbers, for example, "Order number [54321] was shipped on [Tuesday, October 2]." | | | | | | |
|--------------------|---|------------------|---|-------------|--------------------|-------|--|
| Syntax | [Result] = Object. PlayString Value, TermDigits, ClearDigitBuffer, Format, LanguageParams, DateFormat, NumericPrecision | | | | | | |
| Arguments | Name | Data Type | Description | | | | |
| | Value | String | String value to be played. See below for the exact format required by this argument, | | | | |
| | TermDigits | String | Optional. Terminating digit mask. Default is "@" – all digits will terminate playback. | | | | |
| | ClearDigitBuffer | Boolean | Optional. Clear the digit buffer before playback. Default is 1, clear digit buffer. | | | | |
| | Format | FileFormat | Optional. File format of the audio file. For a list of valid file formats, see the FileFormat enumerations for the PlayFile method on page A-20. The default format is 3, 8KHz PCM VOX format. | | | | |
| | LanguageParams | String | Optional. Language parameter. Additional parameter for special language processing. | | | | |
| | DateFormat | String | Optional. See below for the exact format required by this argument, | | | | |
| | NumericPrecision | Integer | Optional. Precision on numbers to be played back. | | | | |
| Returns | Result | ActionResult | Returns information on how the method completed. See the ActionResult enumeration for more information. | | | | |
| Remarks | <p>This method plays the voice string specified by the Value argument. A voice string is specified in the following manner:</p> <pre>"item1[, item2]...[, itemN]"</pre> <p>where N is a maximum of 128 items and the format of an item is:</p> <pre>value element type[format]</pre> <p>Each item consists of three parts:</p> <table> <thead> <tr> <th>Part</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Value</td> <td>Value to be played (for example, "Hello.vox" or "123"). Value can be any valid value for the given element type. Note that for dates, the date value must be in the format of w mm dd hh:nn:ss yyyy</td> </tr> </tbody> </table> | | | Part | Description | Value | Value to be played (for example, "Hello.vox" or "123"). Value can be any valid value for the given element type. Note that for dates, the date value must be in the format of w mm dd hh:nn:ss yyyy |
| Part | Description | | | | | | |
| Value | Value to be played (for example, "Hello.vox" or "123"). Value can be any valid value for the given element type. Note that for dates, the date value must be in the format of w mm dd hh:nn:ss yyyy | | | | | | |

e.g. In Visual Basic's format function you can convert a time value such as Now() to the proper format for the PluginMedia interface as in:

```
Format$(Now(), "w mm dd hh:nn:ss yyyy")
```

Which could return:

```
15 02 08 05:30:00 2001
```

Element type How the file should be played, for example, as a number, or as a currency amount. You can abbreviate any element type by using its first letter. The following element types are available:

| Element type | Abbreviation | Description |
|--------------|--------------|---|
| CHARACTER | C | Speaks the value. |
| DATE | D | Plays the value as a date and/or time in the specified format. |
| FILE | F | Plays the voice file specified by value. |
| INDEXEDFILE | I | Plays a phrase from an indexed voice file. Only one indexed file can be referenced in the string (see below). The indexed file name is specified by the value and the phrase number is specified in the format. |
| MONEY | M | Plays the value as a currency amount (dollars and cents) to the decimal precision specified in the format. |
| NUMBER | N | Plays the value as a number to the decimal precision specified in the format. |
| ORDINAL | O | Plays the value as an ordinal number (1st, 2nd, 3rd, etc.). |

Format Optional. Depending on the element type, format can be any valid date or file format, or precision. Refer to the format table below for valid formats. You only need to specify a format for date, file, indexed file, money, and number element types. The following formats are available:

| Element type | Format |
|--------------|---|
| DATE | A valid date format string to specify how the date is spoken. For example, "15 02 08 05:30:00 2001 D mm/dd/yyyy" plays just the month day and year of a date/time value. To create date values properly, see the value part description above. |
| FILE | A valid file format. For example, "Greeting.vox F 3". |
| INDEXEDFILE | Either an index number or a text string to specify which phrase to play in the indexed file. Text strings must be enclosed in single quotes, for example, "PHRASES.VAP I 5" or "PHRASES.VAP I 'Hello' ". Refer to the PlayIndexedFile method for more information. |
| MONEY | An integer describing decimal precision. For example, "123.50 M 0" will play the money value without any cents. |
| NUMBER | An integer describing decimal precision. For example, "12.345 N 3" will play the number to all 3 decimal places. |

If an error occurs during the playing of any string element, the string halts playing and any remaining elements are not played. Refer to the **PlayFile** method for an explanation of how to terminate this method.

Example 1 The following code plays the string, "Order number 123."

```
MTVMedia.PlayString "ordernum.vox|F,123|C"
```

Example 2 The following code plays the string, "Transferring to..." [John Doe], where the person's name is recorded in a file name stored in the variable Person:

```
MTVMedia.PlayString "transfer.vox|F," & Person & "|F"
```

Example 3 Since only one indexed file can be referenced in the string, so the following syntax would not play both indexed file phrases correctly:

```
MTVMedia.PlayString "file1.vap|I|2, file2.vap|I|2"
```

To work around this limitation, use two PlayString methods:

```
MTVMedia.PlayString "file1.vap|I|2"
```

```
MTVMedia.PlayString "file2.vap|I|2"
```

PlayTone method (PluginMedia interface)

Description Plays a pre-defined tone such as dial tone or a busy signal.

Syntax [Result] = Object.**PlayTone** Tone, TermDigits, ToneTimeOut

Arguments

| Name | Data Type | Description |
|------|-----------|--------------------|
| Tone | ToneID | Tone to be played: |

Enumerations

| <i>Constant</i> | <i>Value</i> | <i>Description</i> |
|----------------------|--------------|-------------------------|
| tidDialtone | 1 | Dialtone |
| tidReorder | 2 | Reorder |
| tidBust | 3 | Busy |
| tidRingback1 | 4 | Ringback 1 |
| tidRingback2 | 5 | Ringback 2 |
| tidRingback1CallWait | 6 | Ringback1 Call Waiting |
| tidRingback2CallWait | 7 | Ringback 2 Call Waiting |
| tidRecallDial | 8 | Recall dial |
| tidIntercept | 9 | Intercept |
| tidCallWait1 | 10 | Call Waiting 1 |
| tidCallWait2 | 11 | Call Waiting 2 |
| tidBusyVerifyA | 12 | Busy Verify 1 |
| tidBusyVerifyB | 13 | Busy Verify 2 |
| tidExecOverride | 14 | Executive override |
| tidFeatureConfirm | 15 | Feature confirmation |
| tidStutterDialtone | 16 | Stutter dialtone |

| | | | |
|----------------|-------------|--------------|---|
| | TermDigits | String | Optional. Terminating digit mask. Default is "@" – all digits will terminate playback. |
| | ToneTimeOut | | Optional. The number of milliseconds before the tone timesout. The default is -1 for no timeout until a digit is pressed. |
| Returns | Result | ActionResult | Returns information on how the method completed. See the ActionResult enumeration for more information |

RecordFile method (PluginMedia interface)

Description Records a voice file, using the specified file format.

Syntax [Result] = Object.**RecordFile** FileName, TermDigits, RecordTimeOut, StartPosition, EndPosition, ClearDigitBuffer, RecordBeep, Format, MaxSilence, MaxNonSilence

| Arguments | Name | Data Type | Description |
|------------------|------------------|------------------|---|
| | FileName | String | File name to be recorded. |
| | TermDigits | String | Optional. Terminating digit mask. Default is "@" – all digits will terminate recording. |
| | RecordTimeOut | Integer | Optional. Maximum number of milliseconds to record. Default is 60000 (1 minute). |
| | StartPosition | Long | Optional. Position in bytes within the file to start recording. Default is 0. |
| | EndPosition | | Optional returned value. Position in bytes within the file where recording stopped. |
| | ClearDigitBuffer | Boolean | Optional. Clear the digit buffer before playback. Default is -1, clear digit buffer. |
| | RecordBeep | Boolean | Optional. Specifies whether a beep will be played prior to recording. Default is -1 (True). |
| | Format | FileFormat | Optional. File format of the audio file. Default is 3, or 8K PCM. See the FileFormat enumeration for a complete list of supported file formats. |
| | MaxSilence | Integer | Optional. Maximum number of milliseconds of silence to record before terminating. Default is -1 or no timeout. |
| | MaxNonSilence | Integer | Optional. Maximum number of milliseconds of non silence to record before terminating. Default is -1 or no timeout |
| Returns | Result | ActionResult | Returns information on how the method completed. See the ActionResult enumeration for more information |

Remarks If the RecordBeep argument is set to True, a "beep" tone is played immediately before recording begins. This method clears the digit buffer before recording unless the AutoClearDigits argument is set to False.

Any of the following conditions can cause the **RecordFile** method to terminate:

A valid terminating digit specified in the TermDigits argument is pressed (or is already in the digit buffer if the AutoClearDigits argument is set to False).

The recording time exceeds the RecordTimeOut argument.

A pause exceeding the MaxSilence argument is detected

A constant sound exceeding the MaxNonSilence argument is detected.

The **Stop** method is invoked.

The caller hangs up.

Example The following example records for up to 20 seconds:

```
mTVMedia.RecordFile "message.vox", , 20000
```

SetRate method (PluginMedia interface)

Description Sets the audio playback rate for the next or current playfile operation.

Syntax `Object.SetRate Rate`

| Arguments | Name | Data Type | Description |
|------------------|-------------|------------------|--|
| | Rate | Integer | Specify a value from -10 to 10 to slow down or speed up voice file playback. |

Remarks -10 is the slowest speed, and 10 is the fastest speed. 0 is normal speed.

Example

SetVolume method (PluginMedia interface)

Description Sets the audio volume for the next or current playfile operation.

Syntax `Object.SetVolume Volume`

| Arguments | Name | Data Type | Description |
|------------------|-------------|------------------|--|
| | Volume | Integer | Specify a value from -10 to 10 to slow down or speed up voice file playback. |

Remarks -10 is the quietest, and 10 is the loudest. 0 is normal volume.

Example

Stop method (PluginMedia interface)

Description Stops all voice processing activity, including PlayFile, PlayString, PlayTone, RecordFile, and GetDigits.

Syntax `Object.Stop`

| Arguments | Name | Data Type | Description |
|------------------|-------------|------------------|--------------------|
| | None. | | |

Remarks Call this method in the CallTerminated method to be sure all voice processing has completed

Example

MIGRATING OLDER IVR PLUG-IN APPLICATIONS

CHAPTER CONTENTS

| | |
|----------------------------------|-----|
| Overview | B-2 |
| IVRPlugInNotify2 interface | B-2 |
| IVRPlugIn2 object | B-4 |

Overview

IVR Plug-in applications written using previous versions of the TeleVantage IVR Plug-in API will continue to run. However we recommend you convert your applications to the current IVR Plug-in interface. This appendix contains a reference to the IVRPlugInNotify2 interface and IVRPlugIn2 object, which have been replaced by the current IVR Plug-in API, described in detail in Appendix A.

IVRPlugInNotify2 interface

The TeleVantage Server invokes the methods of a Notify object to offer calls to the IVR Plug-in or to terminate the IVR Plug-in when the call ends.

Note: The earlier IVRPlugInNotify interface has also been retained for backward compatibility. It is identical to IVRPlugInNotify2 except that it does not include the CallPlaced method.

The IVRPlugInNotify2 interface consists of the following methods:

- **CallOffering method.** See page B-2.
- **CallPlaced method.** See page B-3.
- **CallTerminated method.** See page B-5.

CallOffering method (IVRPlugInNotify2 interface)

Part of IVRPlugInNotify2Interface

Description Invoked when the phone call is offered to the Plug-in. If performing voice processing on the call, you can use the Line argument to allocate the Dialogic voice resource that TeleVantage used to handle the call.

Declaration

```
Private Sub IVRPlugInNotify2_CallOffering( _  
    hParty as Integer, _  
    Line as Integer)
```

| Inputs | Argument | Description |
|----------------|-----------------|--|
| | hParty | Integer value. TeleVantage handle that references the party in this call. |
| | Line | Integer value. Line number which corresponds to the Dialogic voice resource that TeleVantage used to handle this call. |
| Returns | none | |

Remarks Before CallOffering is invoked, TeleVantage creates an instance of the Plug-in's class via its program ID. The program ID is specified in the Administrator's IVR Plug-ins view. You can convert the Line parameter to a standard Dialogic device name using the following pseudo code to calculate the Dialogic board and channel number:

```
Dim c, bb as Integer
Dim DialogicDeviceName as string
If Line Mod 4 = 0 then
    bb = Line \ 4
Else
    bb = Line \ 4 + 1
End If

If Line < 4 then
    c = Line
Else
    If Line \ 4 = 0 then
        c = 4
    Else
        c = Line Mod 4
    End If
End If

DialogicDeviceName = "DXXXB" + bb + "C" + c
```

CallPlaced method (IVRPlugInNotify2 interface)

Description Invoked by the Server to notify the Plug-in when an outbound call has been dialed.

Declaration Private Sub IVRPlugInNotify2_CallPlaced(_
ByVal hParty As Integer, _
ByVal Line As Integer, _
ByVal CallProgress As TVIVRLib.CPType)

| Inputs | Argument | Description |
|---------------|-----------------|---|
| | hParty | Integer value. TeleVantage handle that references the party in this call. |
| | Line | Integer value. Line number, which corresponds to the Dialogic voice resource that TeleVantage used to handle this call. |
| | CallProgress | CPTYPE value. For future use. Currently, this parameter always returns CPDialed. |

Returns none

Remarks TeleVantage does not do any call progress analysis before invoking this method. The number has been dialed on the first available trunk specified by the access code in the PlaceCall method. It is the responsibility of the Plug-in to initiate call progress analysis if required by the Plug-in. Refer to the ReceiveCall and PlaceCall Visual Basic sample projects in the OutBoundCall directory.

CallTerminated method (IVRPlugInNotify2 interface)

Description Invoked when TeleVantage wants to terminate the Plug-in, for example when TeleVantage detects that the caller has hung up.

Declaration

```
Private Sub IVRPlugInNotify2_CallTerminated( _  
    hParty as Integer, _  
    Line as Integer)
```

| Inputs | Argument | Description |
|---------------|-----------------|---|
| | hParty | Integer value. TeleVantage handle that references the party in this call. |
| | Line | Integer value. Line number, which corresponds to the Dialogic voice resource that TeleVantage used to handle this call. See CallOffering (page B-2) for more information. |

Returns none

Remarks CallTerminated is not invoked after the IVRPlugIn CallDone method. After CallTerminated is invoked, TeleVantage removes its reference to the ActiveX EXE's class, which in Visual Basic will then invoke the Class_Terminate subroutine.

IVRPlugIn2 object

The IVR Plug-in application invokes the methods of an IVRPlugIn2 object to exchange call data with the TeleVantage Server, and then to inform the Server when it has finished processing the call.

Note: The earlier IVRPlugIn interface to this component has also been retained for backward compatibility. It is identical to IVRPlugIn2 except that it does not include the PlaceCall and Register methods.

The IVRPlugIn2 object consists of the following methods and properties:

- **CallDone method.** See page B-5.
- **GetCustomPartyData method.** See page B-5.
- **GetPartyData method.** See page B-6.
- **Get XmitTimeslot method.** See page B-7.
- **Listen method.** See page B-7.
- **Ping method.** See page B-7.
- **PlaceCall method.** See page B-8.
- **Register method.** See page B-9.
- **SetCustomPartyData method.** See page B-10.
- **SetCustomData method.** See page B-10.
- **SetStatus method.** See page B-11.
- **Extension property.** See page .B-11
- **DID property.** See page B-11.
- **Name property.** See page B-12.
- **Comments property.** See page B-12.
- **CustomVariableName property.** See page .B-12
- **CustomVariableValue property.** See page B-12.

CallDone method (IVRPlugIn2 object)

Description Returns control of the call to TeleVantage with instructions on how it should continue processing the call.

Syntax `object.CallDone(hParty, NextActionConstants, DialString)`

| Inputs | Argument | Description |
|---------------|---------------------|---|
| | hParty | Integer value. TeleVantage handle that references the party in this call. |
| | NextActionConstants | NextAction value. Specifies action TeleVantage should take on the call via any of the NextAction enumeration constants. |
| | DialString | String value specifying the extension or number that should be used by the NextAction constants NaTransferToNumber and NaTransferToVoiceMail. |

Returns none

Enumerations **NextAction Enumeration**

| <u>Constant</u> | <u>Description</u> |
|-----------------------|--|
| NaHangUp | Hangs up the call |
| NaTransferToNumber | Transfers the call to the extension or external number specified in DialString as if it was dialed from dial tone. |
| NaTransferToVoiceMail | Transfers the call to the voice mail box at the extension specified in DialString. |
| NaReturnToRoutingList | Returns the call to be processed by the routing list it came from, if applicable. For example a user's routing list may be set up to ring extension 102, then call a Plug-in, and then ring extension 103. After the Plug-in accepts the call, performs its processing and calls CallDone with NaReturnToRoutingList, the call will go to extension 103. |

Remarks When this method finishes, the IVR Plug-in is no longer processing the call and the Line argument of the CallOffering method is no longer valid. Also, TeleVantage removes its reference to the ActiveX EXE's class, which in Visual Basic will then invoke the Class_Terminate subroutine. CallTerminated is not invoked after CallDone.

Example

```
'Send the caller to the voice mail box at x102
MTVPlugIn.CallDone(hParty, 2, "102")
```

GetCustomPartyData method (IVRPlugIn2 object)

Description Gets the custom party data from the party in the TeleVantage call. The data that will be retrieved is specified in KeyName. By default calls do not have any custom party data unless an IVR Plug-in has set custom data using the SetCustomPartyData method. The value is returned from the method as a string.

Syntax `PartyData_Str = object.GetCustomPartyData(hParty, KeyName)`

| Inputs | Argument | Description |
|---------------|-----------------|---|
| | hParty | Integer value. TeleVantage handle that references the party in this call. |

KeyName String value specifying what custom data should be retrieved. The KeyName must match exactly a custom data key that was set previously by SetCustomPartyData. If the key does not exist for the party, an empty string "" is returned.

Returns Function result is a string containing the custom party data.

Remarks none

Example

```
'get the caller's CustomerID that was previously set by
'SetCustomPartyData
CustomerID = MTVPPlugIn.GetCustomPartyData(hParty, "CustomerID")
```

GetPartyData method (IVRPlugIn2 object)

Description Gets the standard party data from the party in the TeleVantage call.

Syntax PartyData_Str = object.**GetPartyData**(hParty StandardKeyConstants)

| Inputs | Argument | Description |
|---------------|----------------------|--|
| | hParty | Integer value. TeleVantage handle that references the party in this call. |
| | StandardKeyConstants | KeyType value. Specifies what data should be retrieved from the TeleVantage party via any of the following constants. Note the values may be blank for everything except pdDeviceNumber. |

Returns Function result is a string containing the standard party data for the specified party.

Enumerations KeyType Enumeration

| <u>Constant</u> | <u>Description</u> |
|--------------------------|---|
| pdCallerIDNumber | The Caller ID number |
| pdCallerIDName | The Caller ID name |
| pdDIDNumber | The DID number |
| pdDeviceNumber | If the returned value is negative, the call came in on that trunk number (for example, -5 is trunk #5). If the returned value is positive, the call came in on that internal station number (for example, 5 is station ID 5). |
| pdLoggedInUserExt | The extension of the logged in user. If blank "", then the caller the Plug-in is handling is not a logged in user. This is useful for security purposes. |
| pdCallbackAccessCode | Dialing service access code for a voice message callback number. |
| pdCallbackAddressType | Dialing service address for a voice message callback number. |
| pdCallbackAddressSubType | Dialing service subaddress for a voice message callback number. |
| pdCallbackNumber | Callback number for a voice message. |

Remarks The data that is retrieved is specified in StandardKeyConstants. The value is returned from the method as a string.

Example

```
'get the caller's Caller ID number
CallerIDNumber = MTVPPlugIn.GetPartyData(hParty, 0)
```

GetXmitTimeslot method (IVRPlugIn2 object)

Description Allows the Plug-in to retrieve the SCbus transmit timeslot of the incoming network or station device (MSI, LSI, DTI, or IP). This can be useful to connect a trunk to a GammaLink fax device, Aculab network card or some other device that transmits on a SCbus timeslot.

Syntax Timeslot = object.**GetXmitTimeslot**(hParty)

| Inputs | Argument | Description |
|---------------|-----------------|---|
| | hParty | Integer value. TeleVantage handle that references the party in this call. |

Returns Function result is a Long value containing the transmit timeslot.

Remarks See the Listen method to force TeleVantage to listen to a SCbus timeslot.

Example
'Get the SCBus timeslot that the TeleVantage network or
'station device is transmitting to.
TransmitTimeSlot = MTVPlugIn.GetXmitTimeSlot(hParty)

Listen method (IVRPlugIn2 object)

Description Allows the Plug-in to have the originating Dialogic network or station device (for example, LSI, MSI, DTI, or IP) listen to a specified SCbus timeslot. This can be useful to connect a trunk to a GammaLink fax device, Aculab network card or some other device that transmits on a SCbus timeslot.

Syntax object.**Listen**(hParty, Timeslot)

| Inputs | Argument | Description |
|---------------|-----------------|---|
| | hParty | Integer value. TeleVantage handle that references the party in this call. |
| | Timeslot | Long value. SCbus timeslot that the TeleVantage device should listen to. |

Returns none

Remarks See the GetXmitTimeslot function to get the SCbus timeslot that the TeleVantage network or station device is transmitting to.

Example
'Have the network device listen to SCbus timeslot 5
MTVPlugIn.Listen(hParty, 5)

Ping method (IVRPlugIn2 object)

Description Allows the Plug-in to periodically inform the TeleVantage Server that it is still alive and processing. If the TeleVantage Server does not receive a ping notification after a predefined interval, it will assume that the Plug-in is no longer processing and will reclaim all Dialogic devices associated with it.

Syntax object.**Ping**(hParty)

| Inputs | Argument | Description |
|---------------|-----------------|---|
| | hParty | Integer value. The TeleVantage handle that references the party in this call. |

Returns none

Remarks Ping should be called by the Plug-in every PingInterval minutes, where PingInterval is 2 minutes by default. This allows the Plug-in to periodically inform the TeleVantage Server that it is still alive and processing. If the TeleVantage Server does not receive a ping notification for 2 intervals (by default 4 minutes), TeleVantage will assume that the Plug-in is no longer processing and will reclaim all Dialogic devices associated with it. You can change the PingInterval to be a different value than 2 minutes by adding a PingInterval registry key under the following section:
 HKEY_LOCAL_MACHINE\SOFTWARE\Artisoft\TeleVantage\Server\Settings

It is recommended that you do not Ping in a timer in your Plug-in. Your code may be in an infinite loop, but your timer will continue to fire, telling the TeleVantage Server everything is OK when it is not. Instead spread several "Pings throughout your code whenever lengthy processing occurs such as database access, or prompting the caller for information.

Example

```
'Ping the TeleVantage Server
MTVPlugIn.Ping(hParty)
```

PlaceCall method (IVRPlugIn2 object)

Description Allows the Plug-in to place outbound calls, send data, and return control back to the Server.

Syntax

```
Result% = object.PlaceCall(AccessCode, DialString, _
    Extension, Subtype, InitialData, CallProgress)
```

| Inputs | Argument | Description |
|--------|--------------|--|
| | AccessCode | String value specifying access code for the dialing service used to make the outbound call. |
| | DialString | String value specifying phone number that Server will dial. The format of the number dialed depends on the selected Subtype. See the Subtype enumeration below for examples. |
| | Extension | String value specifying user extension or IVR Plug-in extension to transfer to after dialing is complete. |
| | Subtype | AddressSubType value. Address Subtype for the outbound call. Specify astUseServiceRules to apply dialing service rules to the outbound call. |
| | InitialData | String value specifying a user-defined string to pass to the target Plug-in specified in the Extension parameter above. The string can be retrieved by a Plug-in that handles the call using the GetCustomPartyData method and the "TVIVRInitialData" keyname. |
| | CallProgress | Integer value. Reserved for future use. Currently unused. |

Returns Integer value indicating Device ID of the device on which the call was placed.

Enumerations AddressSubType Enumeration

| <u>Constant</u> | <u>Description</u> |
|------------------|--------------------------|
| astUnknown | Dials the number "as is" |
| astInternational | Reserved for future use |
| astNational | Reserved for future use |
| astNetwork | Reserved for future use |
| astSubscriber | Reserved for future use |
| astAbbreviated | Reserved for future use |

| | |
|---------------------------|--|
| astReserved | Reserved for future use |
| astUseServiceRules | <p>Sends the number through dialing service for processing by dialing rules. The number must be formatted according to one of the following conventions:</p> <p>Option 1 Format: +{country code}{area code and number}</p> <p>Examples: +44 181 5551212 (UK) +1 617 5551212 (US)</p> <p>Option 2 Format: {area code and number}</p> <p>Example: 6175551212</p> |

Remarks none

Example See the PlaceCall and ReceiveCall sample Visual Basic projects in the OutboundCall SDK sample for details on how to use the PlaceCall method. The following Visual Basic code initiates an outbound call:

```
Dim devid As Integer
Dim mServer As IVRPlugIn2
Set mServer = New IVRPlugIn
devid = mServer.PlaceCall("9", "16173540600", "333", 0, _
    "MyInitialDataString", 0)
```

Register method (*IVRPlugIn2 object*)

Description Allows the Plug-in to register with the Server in order to get Server information (Server-side properties) about the Plug-in.

Syntax object.**Register**(ProgramId)

| Inputs | Argument | Description |
|---------------|-----------------|---|
| | ProgramId | String value indicating the Plug-in to be registered. |

Returns Sets values of the Extension, DID, Name, Comments, CustomVariableName, and CustomVariableValue properties.

Remarks When the Register method is invoked, the properties listed above are populated.

Example 'Register the CustomerID sample program.
MTVPlugIn.Register"CustID.CTVCustID"

SetCustomPartyData method (IVRPlugIn2 object)

Description Sets the custom party data for the party in the TeleVantage call. The data that will be set is specified in KeyName.

Syntax `object.SetCustomPartyData(hParty, KeyName, Value)`

| Inputs | Argument | Description |
|--------|----------|--|
| | hParty | Integer value. TeleVantage handle that references the party in this call. |
| | KeyName | String value specifying what custom data should be set. The KeyName will be used later by GetCustomPartyData to retrieve the value from the call. This KeyName is also what must be specified in the Tools-Columns window of the Client's Call Monitor to see the data as a custom column. |
| | Value | String value specifying the value for the custom data specified by the KeyName argument. |

Returns none

Remarks none

Example `'set the caller's CustomerID to 1234567
MTVPlugIn.SetCustomPartyData(hParty, "CustomerID", "1234567")`

SetPartyData method (IVRPlugIn2 object)

Description Sets the standard party data for the party in the TeleVantage call.

Syntax `object.SetPartyData(hParty, StandardKeyConstants, Value)`

| Inputs | Argument | Description |
|--------|----------------------|--|
| | hParty | Integer value. TeleVantage handle that references the party in this call. |
| | StandardKeyConstants | KeyType value. Specifies what data should be set via the KeyType constants pdCallerIDNumber, pdCallerIDName, or pdDIDNumber (note that the pdDeviceNumber constant cannot be used here). |
| | Value | String value specifying the value for the party data specified by the StandardKeyConstants argument. |

Returns none

Enumerations KeyType Enumeration

| <u>Constant</u> | <u>Description</u> |
|-----------------------|--|
| pdCallerIDNumber | The Caller ID number |
| pdCallerIDName | The Caller ID name |
| pdDIDNumber | The DID number |
| pdLoggedInUserExt | The extension of the logged in user. If blank "", then the caller the Plug-in is handling is not a logged in user. This is useful for security purposes. |
| pdCallbackAccessCode | Dialing service access code for a voice message callback number. |
| pdCallbackAddressType | Dialing service address for a voice message callback number. |

| | | |
|----------------|---|---|
| | pdCallbackAddressSubType | Dialing service subaddress for a voice message callback number. |
| | pdCallbackNumber | Callback number for a voice message. |
| Remarks | none | |
| Example | 'set the caller's Caller ID number MTVPlugIn.SetPartyData(hParty, 0, "6175551212") | |

SetStatus method (IVRPlugIn2 object)

Description Sends a string to be displayed in the TeleVantage Device Monitor's status column for the trunk or station that the IVR Plug-in is operating on. Using SetStatus, you can visually monitor a Plug-in's activity. Each call to SetStatus changes the current status to the new status and the old status disappears.

Syntax `object.SetStatus(hParty as Integer, Status as String)`

| Inputs | Argument | Description |
|---------------|-----------------|---|
| | hParty | Integer value. TeleVantage handle that references the party in this call. |
| | Status | String value specifying the new status for the device. |

Returns none

Remarks none

Example 'set the Device Monitor's status for this device
'to the string "Processing caller"
MTVPlugIn.SetStatus(hParty, "Processing caller")

Extension Property (IVRPlugIn2 object)

Description Read-only. Extension number associated with the Plug-in on the Server.

Syntax `Reg_Ext_Str = object.Extension`

Data type string

Remarks Value is set by the Register method.

Example

DID Property (IVRPlugIn2 object)

Description Read-only. DID number associated with the Plug-in on the Server.

Syntax `Reg_DID_Str = object.DID`

Data type string

Remarks Value is set by the Register method.

Example

Name Property (IVRPlugIn2 object)

Description Read-only. Name of the Plug-in on the Server.

Syntax `Reg_Name_Str = object.Name`

Data type string

Remarks Value is set by the Register method.

Example

Comments Property (IVRPlugIn2 object)

Description Read-only. Comments associated with the Plug-in on the Server.

Syntax `Reg_Comments_Str = object.Comments`

Data type string

Remarks Value is set by the Register method.

Example

CustomVariableName Property (IVRPlugIn2 object)

Description Read-only. User-defined variable name associated with the Plug-in on the Server.

Syntax `Reg_VarName_Str = object.CustomVariableName`

Data type string

Remarks Value is set by the Register method.

Example

CustomVariableValue Property (IVRPlugIn2 object)

Description Read-only. User-defined variable value for the variable name associated with the Plug-in on the Server.

Syntax `Reg_VarVal_Str = object.CustomVariableValue`

Data type string

Remarks Value is set by the Register method.

Example

DEVICE STATUS API REFERENCE

CHAPTER CONTENTS

Server object

| | |
|--|-----|
| Connect method (Server object) | C-2 |
| Disconnect method (Server object) | C-2 |
| Devices Property (Server object) | C-2 |
| DeviceStateChanged Event (Server object) | C-3 |

Device object

| | |
|---|-----|
| Refresh method (Device object) | C-3 |
| AssignedExtensions Property (Device object) | C-3 |
| Class Property (Device object) | C-4 |
| CurrentExtension Property (Device object) | C-4 |
| DefaultExtension Property (Device object) | C-4 |
| HookState Property (Device object) | C-5 |
| Name Property (Device object) | C-5 |
| Number Property (Device object) | C-5 |
| Status Property (Device object) | C-6 |

Extension object

| | |
|---|-----|
| Refresh method (Extension object) | C-7 |
| ACDDoNotDisturb Property (Extension object) | C-7 |
| DoNotDisturb Property (Extension object) | C-7 |
| Extension Property (Extension object) | C-7 |
| FirstName Property (Extension object) | C-8 |
| LastName Property (Extension object) | C-8 |

Overview

This appendix describes the methods, properties, and events that make up the TeleVantage Device Status API. For more information about using the Device Status API, see Chapter 4.

Server object

Connect method (Server object)

Description Connects to the specified server and initializes the Server object.

Syntax `GoodConnect = object.Connect(ServerName)`

| Inputs | Argument | Description |
|---------------|-----------------|---------------------------------------|
| | ServerName | The name of the server to connect to. |

Returns Boolean value indicating if the connection was successful.

Remarks none

Example See the DeviceStatusSample project's cbConnect_Click subroutine.

Disconnect method (Server object)

Description Disconnects from the server.

Syntax `object.Disconnect()`

Inputs none

Returns none

Remarks none

Example See the DeviceStatusSample project's cbConnect_Click and Form_Unload subroutines.

Devices Property (Server object)

Description Read only. Returns a collection of Device objects representing all devices on the current server.

Syntax `DeviceList = object.Devices`

Data type **Collection of Device objects**

Remarks none

Example See the DeviceStatusSample project's RefreshDisplay subroutine.

DeviceStateChanged Event (Server object)

| | | |
|--------------------|---|---------------------------------|
| Description | Raised whenever the state of the device changes. | |
| Syntax | <code>DeviceStateChanged(ByVal Device As Device)</code> | |
| Inputs | none | |
| Returns | Argument | Description |
| | Device | The Device object that changed. |
| Remarks | none | |
| Example | See the DeviceStatusSample project's mTVServer_DeviceStateChanged subroutine. | |

Device object

Refresh method (Device object)

| | |
|--------------------|---|
| Description | Refreshes information about the extensions on the device. |
| Syntax | <code>object.Refresh()</code> |
| Inputs | none |
| Returns | none |
| Remarks | none |

AssignedExtensions Property (Device object)

| | |
|--------------------|---|
| Description | Read only. Returns a collection of Extension objects representing all extensions assigned to this device. |
| Syntax | <code>ExtensionList = object.AssignedExtensions</code> |
| Data type | Collection of Extension objects |
| Remarks | none |

Class Property (Device object)

Description Read only. Returns the device's class (station or trunk).

Syntax `TVObjectDeviceClass = object.Class`

Data type **Long** (TVObjectDeviceClass enumeration)

Enumerations **TVObjectDeviceClass Enumeration**

| <u>Constant</u> | <u>Description</u> |
|-----------------------------------|----------------------|
| <code>tvDeviceClassTrunk</code> | Device is a trunk. |
| <code>tvDeviceClassStation</code> | Device is a station. |

Remarks none

CurrentExtension Property (Device object)

Description Read only. Returns the Extension object currently logged in to the device.

Syntax `Extension = object.CurrentExtension`

Data type **Extension object**

Remarks Returns Nothing if there is no current extension.

Example See the DeviceStatusSample project's UpdateDevice subroutine.

DefaultExtension Property (Device object)

Description Read only. Returns the Extension object that is the default for the device.

Syntax `Extension = object.DefaultExtension`

Data type **Extension object**

Remarks Returns Nothing if there is no default extension.

Example See the DeviceStatusSample project's UpdateDevice subroutine.

HookState Property (Device object)

| Description | Read only. Returns the hook state of the device. | | | | | | | | |
|---------------------------------------|---|-----------------|--------------------|----------------------------------|--------------------|-----------------------------------|---------------------|---------------------------------------|--|
| Syntax | <code>TVDeviceHookState = object.HookState</code> | | | | | | | | |
| Data type | Long (TVDeviceHookState enumeration) | | | | | | | | |
| Enumerations | TVDeviceHookState Enumeration | | | | | | | | |
| | <table><thead><tr><th><u>Constant</u></th><th><u>Description</u></th></tr></thead><tbody><tr><td><code>tvDeviceHookStateOn</code></td><td>Device is on-hook.</td></tr><tr><td><code>tvDeviceHookStateOff</code></td><td>Device is off-hook.</td></tr><tr><td><code>tvDeviceHookStateUnknown</code></td><td>Device's hook state can't be determined.</td></tr></tbody></table> | <u>Constant</u> | <u>Description</u> | <code>tvDeviceHookStateOn</code> | Device is on-hook. | <code>tvDeviceHookStateOff</code> | Device is off-hook. | <code>tvDeviceHookStateUnknown</code> | Device's hook state can't be determined. |
| <u>Constant</u> | <u>Description</u> | | | | | | | | |
| <code>tvDeviceHookStateOn</code> | Device is on-hook. | | | | | | | | |
| <code>tvDeviceHookStateOff</code> | Device is off-hook. | | | | | | | | |
| <code>tvDeviceHookStateUnknown</code> | Device's hook state can't be determined. | | | | | | | | |
| Remarks | none | | | | | | | | |
| Example | See the DeviceStatusSample project's UpdateDevice subroutine. | | | | | | | | |

Name Property (Device object)

| | |
|--------------------|---|
| Description | Read only. Returns the name of the device (blank for stations and the trunk name for trunks). |
| Syntax | <code>NameString = object.Name</code> |
| Data type | String |
| Remarks | none |
| Example | See the DeviceStatusSample project's UpdateDevice subroutine. |

Number Property (Device object)

| | |
|--------------------|---|
| Description | Read only. Returns the number of the device (positive for stations and negative for trunks). |
| Syntax | <code>DevNumber = object.Number</code> |
| Data type | Long |
| Remarks | none |
| Example | See the DeviceStatusSample project's mTVServer_DeviceStateChanged and RefreshDisplay subroutines. |

Status Property (Device object)**Description** Read only. Returns a status code indicating the device's current activity.**Syntax** `TVDeviceStatus = object.Status`**Data type** **Long** (TVDeviceStatus enumeration)**Enumerations TVDeviceStatus Enumeration**

| <u>Constant</u> | <u>Description</u> |
|---|---|
| <code>tvDeviceStatusAudioCtl</code> | Station is playing/recording audio for the Client. |
| <code>tvDeviceStatusCallingStation</code> | Station or trunk is waiting while TeleVantage rings a station. |
| <code>tvDeviceStatusDialogNode</code> | Station or trunk is navigating an auto-attendant or routing list. |
| <code>tvDeviceStatusExtDialing</code> | Trunk is dialing an external number. |
| <code>tvDeviceStatusHoldRecall</code> | Station is calling back after a hold timed out. |
| <code>tvDeviceStatusIdle</code> | Station or trunk is idle. |
| <code>tvDeviceStatusInCall</code> | Station or trunk has a call in progress. |
| <code>tvDeviceStatusIncomingCall</code> | Station or trunk is receiving an incoming call. |
| <code>tvDeviceStatusIntDialing</code> | Station is dialing an internal number. |
| <code>tvDeviceStatusListenCall</code> | Station or trunk is screening a message. |
| <code>tvDeviceStatusLoggingIn</code> | Station or trunk is logging into the TUI. |
| <code>tvDeviceStatusNoLC</code> | Trunk does not have loop current. |
| <code>tvDeviceStatusOfferingCall</code> | Station or trunk is offering a call. |
| <code>tvDeviceStatusOutboundDialing</code> | Station or trunk is dialing an external number. |
| <code>tvDeviceStatusPlayingHoldMusic</code> | Station is playing hold music. |
| <code>tvDeviceStatusPreIdle</code> | Station or trunk is in a pre-idle state. |
| <code>tvDeviceStatusReorder</code> | Station has been left off-hook too long, or trunk is in an error condition. |
| <code>tvDeviceStatusStartup</code> | Station or trunk is waiting while the TeleVantage system starts up. |
| <code>tvDeviceStatusTakingMsg</code> | Station or trunk is taking a message. |
| <code>tvDeviceStatusTui</code> | Station or trunk is navigating the TUI. User is listening to the auto-attendant menu. |
| <code>tvDeviceStatusUnknown</code> | Station or trunk is not responding to device status queries. |
| <code>tvDeviceStatusDBByPass</code> | Device is functioning, but Server database is not available. |

Remarks none**Example** See the DeviceStatusSample project's UpdateDevice subroutine.

Extension object

Refresh method (Extension object)

Description Refreshes information about this extension.

Syntax `object.Refresh()`

Inputs none

Returns none

Remarks none

ACDDoNotDisturb Property (Extension object)

Description Read only. Returns a boolean value indicating whether or not the extension is in ACD Do Not Disturb state.

Syntax `UserACD_DND = object.ACDDoNotDisturb`

Data type Boolean

| Setting | Description |
|----------------|---|
| True | Extension is in ACD Do Not Disturb state. |
| False | Extension is not in ACD Do Not Disturb state. |

Remarks none

DoNotDisturb Property (Extension object)

Description Read only. Returns a boolean value indicating whether or not the extension is in Do Not Disturb state.

Syntax `UserDND = object.DoNotDisturb`

Data type Boolean

| Setting | Description |
|----------------|---|
| True | Extension is in Do Not Disturb state. |
| False | Extension is not in Do Not Disturb state. |

Remarks none

Extension Property (Extension object)

Description Read only. Returns the extension number for this user.

Syntax `UserExtensionString = object.Extension`

Data type String

Remarks none

Example See the DeviceStatusSample project's UpdateDevice subroutine.

FirstName Property (Extension object)

| | |
|--------------------|--|
| Description | Read only. Returns the first name of the user on this extension. |
| Syntax | UserFirstNameString = object. FirstName |
| Data type | String |
| Remarks | none |
| Example | See the DeviceStatusSample project's UpdateDevice subroutine. |

LastName Property (Extension object)

| | |
|--------------------|---|
| Description | Read only. Returns the last name of the user on this extension. |
| Syntax | UserLastNameString = object. LastName |
| Data type | String |
| Remarks | none |
| Example | See the DeviceStatusSample project's UpdateDevice subroutine. |

CLIENT API OBJECT REFERENCE

CHAPTER CONTENTS

| | |
|--------------------------------------|------|
| Address object | D-3 |
| Addresses object | D-3 |
| Agent object | D-4 |
| Agents object | D-5 |
| AgentStat object | D-5 |
| AudioClip object | D-6 |
| Call object | D-7 |
| CallHistory object | D-9 |
| CallRule object | D-10 |
| Column object | D-11 |
| Columns object | D-11 |
| Contact object | D-12 |
| Error object | D-13 |
| Field object | D-13 |
| Fields object | D-13 |
| Folder object | D-14 |
| Folders object | D-15 |
| Greeting object | D-15 |
| IAssociate object | D-16 |
| ICOMSecurity object | D-16 |
| IDialingService object | D-17 |
| IItem object | D-17 |
| InternetService object | D-18 |
| IPhoneService object | D-18 |
| Items object | D-19 |
| LocaleCode object | D-19 |
| LocaleCodes object | D-20 |
| Message object | D-20 |
| Notification object | D-21 |
| NotifyScheduleItem object | D-21 |
| NotifyScheduleItems object | D-22 |
| Parties object | D-22 |
| Party object | D-23 |
| PartyHistories object | D-24 |
| PartyHistory object | D-24 |
| Permission object | D-25 |
| Permissions object | D-25 |
| PersonalStatus object | D-26 |
| PhoneGatewayService object | D-27 |
| PhoneService object | D-28 |
| QueueByPeriodStat object | D-29 |

| | |
|---|------|
| QueueByShiftStat object | D-30 |
| QueueStat object | D-30 |
| Recipients object | D-32 |
| Role object | D-32 |
| Roles object | D-32 |
| RoutingList object | D-33 |
| RoutingListAction object | D-34 |
| RoutingListActions object | D-34 |
| RoutingListFinalAction object | D-35 |
| RoutingService object | D-35 |
| Schedule object | D-36 |
| ScheduledDate object | D-36 |
| ScheduledDates object | D-36 |
| ScheduledDay object | D-37 |
| ScheduledDays object | D-37 |
| Schedules object | D-38 |
| Session object | D-38 |
| Shortcut object | D-43 |
| ShortcutGroup object | D-44 |
| ShortcutGroups object | D-44 |
| Shortcuts object | D-45 |
| Station object | D-45 |
| StationButton object | D-46 |
| StationButtons object | D-46 |
| StationFeature object | D-47 |
| StationFeatures object | D-47 |
| StationParameter object | D-48 |
| StationParameters object | D-48 |
| StationType object | D-48 |
| StationTypes object | D-49 |
| SwitchGatewayService object | D-49 |
| SwitchService object | D-50 |
| System object | D-50 |
| SystemSetting object | D-51 |
| SystemSettings object | D-52 |
| SystemTarget object | D-52 |
| User object | D-53 |
| View object | D-55 |
| Views object | D-56 |
| Workgroup object | D-56 |
| WorkgroupMember object | D-57 |
| WorkgroupMembers object | D-57 |

Address object

Contains address information for a contact. Address objects can be collected in an Addresses object.

Methods:

GetCallerIDMatch([AddressID]) **as Contact**

AddressID as String. Optional. Default value is "".

GetPhoneNumberComponent(Component) **as Variant**

Component as TVAddressPhoneComponent.

GuessAddressType(AddressString) **as TVAddressType**

AddressString as String.

Validate([AllowOverride]) **as TVAddressError**

AllowOverride as Boolean. Optional. Default value is 0 (false).

Properties:

AccountCode as .

AddressSubType as .

AddressType as .

AssociatedPersonID as .

Category as .

Default as .

Description as .

ID as .

IsPager as .

Parent as .

Service as .

Session as .

Tag as .

Usage as .

UseRules as .

UnformattedValue as .

Value as .

Addresses object

A collection of Address objects.

Methods:

Add([Value], [AddressType], [AddressSubType], [Service], [Category]) **as Address**

Value as String. Optional.

AddressType as TVAddressType. Optional. Default value is 1.

AddressSubType as TVAddressSubType. Optional.

Service as IDialingService. Optional.

Category as TVAddressCategory. Optional.

Discard()

Exists(Index, [SearchType]) **as Boolean**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) **as Address**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Remove(Index, [SearchType])

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Restrict(Category, [CreatelfNecessary], [Value], [AddressType], [AddressSubType], [Service]) **as Addresses**

Category as TVAddressCategory.

CreatelfNecessary as Boolean. Optional. Default value is 0 (false).

Value as String. Optional.

AddressType as TVAddressType. Optional. Default value is 1.

AddressSubType as TVAddressSubType. Optional.

Service as IDialingService. Optional.

Save()

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

Agent object

Contains information about a Call Center agent. Agent objects can be collected in an Agents object or in a folder of type tvFolderAgents (see "Folder types" on page 2-10). See "Agent objects" on page 2-11 for a detailed description of Agent object structure.

Methods:

Discard()

EndWrapUp()

SetObserver(Observer)

Observer as Boolean.

Properties:

Class as .

HiddenObserver as .

ID as .

Name as .

Observer as .

Parent as .

Permissions as .

QueueID as .

Session as .

Synchronized as .

Tag as .

Agents object

A collection of Agent objects.

Methods:

Exists(Index, [SearchType]) **as Boolean**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) **as Agent**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

AgentStat object

Contains information concerning a Call Center agent's activity. AgentStat objects can be collected in a folder of type tvFolderAgentStats (see "Folder types" on page 2-10).

Methods:

SetObserver(Observer)

Observer as Boolean.

Properties:

ActiveDurationTotalInbound as .

ActiveDurationTotalOutbound as .

AgentID as .

Class as .

ID as .

ItemsActiveTotalInbound as .

ItemsActiveTotalOutbound as .

LongestItemDurationInbound as .

LongestTalkTimeInbound as .

LongestTalkTimeOutbound as .

LongestWrapupTimeInbound as .

Observer as .

Overflow as .

Parent as .

Position as .

QueueID as .

Session as .

Status as .

StatusStartTime as .

Synchronized as .

Tag as .

UserID as .

Away as .

ForcedBreakCount as .
ItemsCompletedTotalOutbound as .
ItemsConnectedTotalOutbound as .
ItemsPlacedTotalOutbound as .
LongestItemDurationOutbound as .
LongestWrapupTimeOutbound as .
NoAnswerCount as .
TalkTimeTotalOutbound as .
WrapupTimeTotalOutbound as .

Events:

Change()
Remove()

AudioClip object

Used to create or manipulate an audio file.

Methods:

Export(ExportFile, [VoxFormat])
ExportFile as String.
VoxFormat as TVAudioVOXFormat. Optional. Default value is -1.
Import(ImportFile, [Insert], [VoxFormat])
ImportFile as String.
Insert as Boolean. Optional. Default value is 0 (false).
VoxFormat as TVAudioVOXFormat. Optional. Default value is -1.
Pause()
Play([StartPos], [StopPos])
StartPos as Long. Optional. Default value is 0.
StopPos as Long. Optional. Default value is -1.
Record([StartPos], [Insert])
StartPos as Long. Optional. Default value is 0.
Insert as Boolean. Optional. Default value is 0 (false).
ResumeAfterPause()
StopAudio()

Properties:

BookMarkStart as .
BookMarkEnd as .
CurrentPosition as .
DeviceType as .
Duration as .
FileName as .
ID as .
Parent as .
PreventHybrid as .
ReadOnly as .
Session as .
Size as .
SkipInterval as .
State as .

Tag as .

Events:

AudioDeviceChange(OldDevice, NewDevice)

OldDevice as TVAudioDeviceType.

NewDevice as TVAudioDeviceType.

PlayBeforeInterrupt(ShutdownDelay)

ShutdownDelay as Long.

RecordBeforeInterrupt(ShutdownDelay)

ShutdownDelay as Long.

SkipIntervalChange(OldValue, NewValue)

OldValue as Long.

NewValue as Long.

StateChange(FromState, ToState, Reason)

FromState as TVAudioState.

ToState as TVAudioState.

Reason as TVAudioStateChangeReason.

Error(ErrorNumber)

ErrorNumber as Long.

CopyAudioFromServer()

Call object

Contains information about a call and methods to manipulate the call. Call objects can be collected in a folder of type tvFolderCalls (see "Folder types" on page 2-10).

Methods:

Announce([AnnounceType])

AnnounceType as TVCallAnnounceType. Optional. Default value is 1.

Answer()

Associate(Contact)

Contact as Contact.

BlindTransfer(Address)

Address as Address.

Conference(ConferenceCall) as Call

ConferenceCall as Call.

Decline([DeclineMode], [DeclineAudio])

DeclineMode as TVCallDeclineMode. Optional. Default value is 0.

DeclineAudio as AudioClip. Optional.

Disconnect()

Hold([HoldAudioType], [HoldAudio])

HoldAudioType as TVCallGrabAndHoldAudioType. Optional. Default value is 2.

HoldAudio as AudioClip. Optional. Default value is <unprintable IDispatch*>.

Join([Role], [Partner])

Role as TVPartyRole. Optional. Default value is 0.

Partner as Party. Optional. Default value is <unprintable IDispatch*>.

Park() as Long

RecordPause()

RecordResume()

RecordStart([Target], [Format], [TimeOut], [TermDigits], [Beep])

Target as IItem. Optional. Default value is <unprintable IDispatch*>.

Format as TVCallRecordFormat. Optional. Default value is 3.

TimeOut as Long. Optional. Default value is -1.

TermDigits as String. Optional. Default value is "".

Beep as Boolean. Optional. Default value is 0 (false).

RecordStop()

SetAccountCode(AccountCode)

AccountCode as String.

SetCustomData(Key, Value)

Key as String.

Value as String.

SetRole(Role, [Partner])

Role as TVPartyRole.

Partner as Party. Optional. Default value is <unprintable IDispatch*>.

SupervisedTransferAbort()

SupervisedTransferComplete()

SupervisedTransferConference() as Call

SupervisedTransferStart(Address) as Call

Address as Address.

Supports(Feature) as Boolean

Feature as TVCallFeature.

Unpark() as Call

Properties:

AccountCode as .

Address as .

AnsweredByDevice as .

AnsweredByFirstName as .

AnsweredByLastName as .

AnsweredByName as .

AnsweredByUserID as .

AnsweredTime as .

AssociatedPerson as .

AssociatedPersonID as .

AssociatedPersonType as .

AudioClip as .

CallType as .

Class as .

Comments as .

CustomData as .

Device as .

DID as .

Direction as .

Features as .

FirstName as .

ID as .

LastName as .

Mute as .

Name as .

Number as .

OrbitNumber as .

OwnerAddress as .
OwnerDevice as .
OwnerFirstName as .
OwnerID as .
OwnerLastName as .
OwnerName as .
OwnerParty as .
OwnerPartyID as .
Parent as .
Parties as .
PartyCount as .
Priority as .
RecordStatus as .
Role as .
ScreenMessage as .
Session as .
StartTime as .
Status as .
Synchronized as .
Tag as .
Handle as .
LineAppearance as .
WaitStartTime as .

Events:

Change(Synchronized)
Synchronized as Boolean.
PermissionsChange()
Remove()

CallHistory object

Contains information about a call, similar to what is displayed in the Client's Call Log view. CallHistory objects can be collected in a folder of type tvFolderCallHistory (see "Folder types" on page 2-10). See "CallHistory objects" on page 2-12 for a detailed description of the CallHistory object structure.

Methods:

Associate(Contact)
Contact as Contact.
Delete([Mode])
Mode as TVDeleteMode. Optional. Default value is 0.
Discard()
Save([Overwrite])
Overwrite as Boolean. Optional. Default value is 0 (false).
tvGetAssociatedPersonID() as String

Properties:

AssociatedPerson as .
AssociatedPersonID as .
AssociatedPersonType as .
CallbackAddress as .

Class as .
Comments as .
Direction as .
EndTime as .
FromParty as .
ID as .
Parent as .
PartyHistories as .
Result as .
Session as .
StartTime as .
Synchronized as .
Tag as .
ToParty as .
WaitTime as .
AccountCode as .
OrganizationName as .
QueueRecordingMessage as .
QueueRecordingMessageID as .
UserRecordingMessage as .
UserRecordingMessageID as .

Events:

Change(Synchronized, ChangedBy)
 Synchronized as Boolean.
 ChangedBy as TVItemChangedBy.
Remove()

CallRule object

Contains information about a call rule, similar to what is displayed in the Client's Call Rules view. CallRule objects can be collected in a folder of type tvFolderCallRules (see "Folder types" on page 2-10). See "CallRule objects" on page 2-13 for a detailed description of the CallRule object structure.

Methods:

Delete([Mode])
 Mode as TVDeleteMode. Optional. Default value is 0.
Discard()
Save([Overwrite])
 Overwrite as Boolean. Optional.
SwapPriorities(OtherCallRule)
 OtherCallRule as CallRule.

Properties:

CallerCondition as .
CallerTypeCondition as .
Class as .
CustomScheduleCondition as .
Enabled as .
Greeting as .
ID as .

Name as .
Parent as .
PersonalStatus as .
Priority as .
RingOverride as .
RoutingList as .
ScheduleTypeCondition as .
Session as .
Synchronized as .
Tag as .

Events:

Change(Synchronized, ChangedBy)
 Synchronized as Boolean.
 ChangedBy as TVItemChangedBy.
Remove()

Column object

Contains information about columns in a Client view. Column objects can be collected in a Columns object. See "The Folder object" on page 2-8 for details about Folder object structures.

Properties:

Class as .
DataType as .
DisplayName as .
ID as .
Name as .
Parent as .
Position as .
ResourceID as .
Session as .
Tag as .
Width as .

Columns object

A collection of Column objects.

Methods:

Add(Name) as **Column**
 Name as String.
Discard()
Exists(Index, [SearchType]) as **Boolean**
 Index as Variant.
 SearchType as TVSearchType. Optional. Default value is 0.
Item(Index, [SearchType]) as **Column**
 Index as Variant.
 SearchType as TVSearchType. Optional. Default value is 0.
Remove(Index, [SearchType])
 Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Save()

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

Contact object

Contains information about a contact, similar to the information displayed in the Client's Contacts view. Contact objects can be collected in a WorkgroupMembers object or in a folder of type tvFolderContacts (see "Folder types" on page 2-10). See "Contact objects" on page 2-13 for a detailed description of the Contact object structure.

Methods:

Associate(Item, AssociateFlags) as Boolean

Item as IAssociate.

AssociateFlags as TVContactAssociateFlags.

Delete([Mode])

Mode as TVDeleteMode. Optional. Default value is 0.

Discard()

Move(Destination)

Destination as Folder.

Save([Overwrite])

Overwrite as Boolean. Optional.

Properties:

AccountCode as .

Addresses as .

Class as .

Comments as .

Company as .

DefaultAddress as .

FirstName as .

FullName as .

ID as .

JobTitle as .

LastName as .

Parent as .

PIN as .

Session as .

Synchronized as .

Tag as .

TelephonePromptLanguage as .

VoiceTitle as .

Events:

Change(Synchronized, ChangedBy)

Synchronized as Boolean.

ChangedBy as TVItemChangedBy.

Move()

Remove()

Error object

Retains a record of a system error message.

Methods:

Clear()

Properties:

Description as .

Number as .

source as .

Field object

Contains information about fields in a Client view. Field objects can be collected in a Fields object. See "The Folder object" on page 2-8 for details about Folder object structures.

Properties:

Class as .

DataType as .

DisplayName as .

ID as .

Name as .

Parent as .

ResourceID as .

Session as .

Tag as .

Fields object

A collection of Field objects. See "The Folder object" on page 2-8 for details about Folder object structures.

Methods:

Exists(Index, [SearchType]) as **Boolean**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) as **Field**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

Folder object

Used to organize most of the objects available in a Session. Folder objects can be collected in a Folders object. See "Structure of a session" on page 2-4 and "How folders are used" on page 2-7 for details about folders.

Methods:

Delete([Mode])

Mode as TVDeleteMode. Optional. Default value is 0.

Discard()

GetMailboxDiskUsage(CurrentUsage, MaximumUsage)

CurrentUsage as Long.

MaximumUsage as Long.

GetOwner() as SystemTarget

GetProperty(Name, [Default]) as String

Name as String.

Default as String. Optional.

GetUnheardCount() as Long

Move(Destination)

Destination as Folder.

Save([Overwrite])

Overwrite as Boolean. Optional. Default value is 0 (false).

SetProperty(Name, Value)

Name as String.

Value as String.

Properties:

AccessLevel as .

Class as .

CurrentView as .

Customizable as .

Default as .

DisplayName as .

Fields as .

Folders as .

Hidden as .

ID as .

ItemClass as .

Items as .

Name as .

OwnerID as .

Parent as .

Permissions as .

Session as .

Synchronized as .

Tag as .

Views as .

Events:

Change(Synchronized, ChangedBy)

Synchronized as Boolean.

ChangedBy as TVItemChangedBy.

Move()

Remove()

Folders object

A collection of Folder objects.

Methods:

Add(Name, [ItemClass]) **as Folder**

Name as String.

ItemClass as TVObjectItemClass. Optional. Default value is -1.

Exists(Index, [SearchType]) **as Boolean**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) **as Folder**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Remove(Index, [SearchType])

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Restrict(Filter) **as Folders**

Filter as String.

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

Greeting object

Contains information about a greeting, similar to the information displayed in the Client's Greetings view. Greeting objects can be collected in a folder of type tvFolderGreetings (see "Folder types" on page 2-10). See "Greeting objects" on page 2-15 for a detailed description of the Greeting object structure.

Methods:

Delete([Mode])

Mode as TVDeleteMode. Optional. Default value is 0.

Discard()

Save([Overwrite])

Overwrite as Boolean. Optional. Default value is 0 (false).

Properties:

Active as .

AudioClip as .

Class as .

Default as .

ID as .

LastRecordedTime as .

Name as .

Parent as .
Session as .
Synchronized as .
Tag as .
Text as .

Events:

Change(Synchronized, ChangedBy)
 Synchronized as Boolean.
 ChangedBy as TVItemChangedBy.
Remove()

IAssociate object

Generic interface used to communicate with objects that can be associated. You do not need to deal directly with an IAssociate interface; rather, you work with the associated object itself.

Methods:

Associate(Contact)
 Contact as Contact.
Save([Overwrite])
 Overwrite as Boolean. Optional. Default value is 0 (false).

Properties:

AssociateAllowed as .
AssociatedPerson as .
AssociatedPersonID as .
AssociatedPersonType as .
CallbackAddress as .
CallerIDAddress as .
CallerIDName as .
CallerIDNumber as .
Class as .
ID as .
VoiceTitle as .

ICOMSecurity object

TBD

Methods:

OverrideSecurity(EXENAME, AppID, FriendlyName) **as Boolean**
 EXENAME as String.
 AppID as String.
 FriendlyName as String.
RestoreSecurity(EXENAME, [AppID])
 EXENAME as String.
 AppID as String. Optional. Default value is "".

Properties:

UseSecurityBypass as .

IDialingService object

Generic interface used to communicate with any of the dialing service objects. You do not need to deal directly with an IDialingService interface; rather, you work with the dialing service object itself. See "Dialing Service objects" on page 2-14 for information about all of the dialing service objects.

Properties:

AccessCode as .
Name as .
Class as .
ID as .
AddressType as .
Hidden as .

IItem object

Generic interface used to communicate with item objects. You do not need to deal directly with an IItem interface; rather, you work with the item object itself.

Methods:

CheckAccess(AccessLevel)
AccessLevel as TVPermissionAccessLevel.
Copy([Parent]) as IItem
Parent as Folder. Optional.
Delete([Mode])
Mode as TVDeleteMode. Optional. Default value is 0.
Discard()
Initialize(Parent, Data, Session)
Parent as Folder.
Data as Recordset.
Session as Session.
Move(Destination)
Destination as Folder.
Save([Overwrite])
Overwrite as Boolean. Optional. Default value is 0 (false).

Properties:

Class as .
Comments as .
DerivedClass as .
DisplayName as .
ID as .
Name as .
Parent as .
Session as .
Status as .
Synchronized as .

InternetService object

A dialing service object containing information for dialing a number over the Internet. InternetService objects can be collected in a folder of type tvFolderServices (see "Folder types" on page 2-10). See "Dialing Service objects" on page 2-14 for information about all of the dialing service objects.

Methods:

Delete([Mode])

Mode as TVDeleteMode. Optional. Default value is 0.

Discard()

Move(Destination)

Destination as Folder.

Save([Overwrite])

Overwrite as Boolean. Optional.

Properties:

AccessCode as .

AddressType as .

Class as .

Default as .

Hidden as .

ID as .

Name as .

Parent as .

Session as .

Synchronized as .

Tag as .

Events:

Change(Synchronized, ChangedBy)

Synchronized as Boolean.

ChangedBy as TVItemChangedBy.

Remove()

IPhoneService object

Generic interface used to communicate with PhoneService, PhoneGatewayService, and RoutingService objects. You do not need to deal directly with an IPhoneService object; rather, you work with the dialing service object itself. See "Dialing Service objects" on page 2-14 for information about all of the dialing service objects.

Properties:

AreaCode as .

CountryCode as .

IncludeLongDistancePrefix as .

InternationalPrefix as .

LongDistancePrefix as .

Items object

A collection of Item objects.

Methods:

Add([Class]) as IDispatch

Class as TVObjectClass. Optional. Default value is -1.

Exists(Index, [SearchType]) as Boolean

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Find(Filter) as IDispatch

Filter as String.

FindNext() as IDispatch

Item(Index, [SearchType]) as IDispatch

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Remove(Index, [SearchType])

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Restrict(Filter) as Items

Filter as String.

Sort(Field, [SortOrder])

Field as String.

SortOrder as TVSortOrder. Optional. Default value is 0.

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

LocaleCode object

Contains a locale code identifying a TeleVantage language locale setting. LocaleCode objects can be collected in a LocaleCodes object.

Properties:

Description as .

LocaleCode as .

Parent as .

Session as .

Tag as .

LocaleCodes object

A collection of LocaleCode objects.

Methods:

Exists(Index, [SearchType]) **as Boolean**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) **as LocaleCode**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

Message object

Contains a voice message and related information. Message objects can be collected in a folder of type tvFolderMessages (see "Folder types" on page 2-10). See "Message objects" on page 2-15 for a detailed description of the Message object structure.

Methods:

Associate(Contact)

Contact as Contact.

Delete([Mode])

Mode as TVDeleteMode. Optional. Default value is 0.

Discard()

Forward() **as Message**

Move(Destination)

Destination as Folder.

Reply([ReplyToAll]) **as Message**

ReplyToAll as Boolean. Optional. Default value is 0 (false).

Save([Overwrite])

Overwrite as Boolean. Optional. Default value is 0 (false).

Send() **as Recipients**

Properties:

AssociatedPerson as .

AssociatedPersonID as .

AssociatedPersonType as .

AudioClip as .

CallbackAddress as .

Class as .

Comments as .

Confidential as .

CreatedTime as .

ForwardedTime as .

ID as .

MessageType as .
Parent as .
ReceiveStatus as .
Recipients as .
SenderCallerIDAddress as .
SenderCallerIDName as .
SenderCallerIDNumber as .
SendStatus as .
Session as .
Synchronized as .
Tag as .
Unheard as .
Urgent as .
CallHistoryID as .

Events:

Change(Synchronized, ChangedBy)
 Synchronized as Boolean.
 ChangedBy as TVItemChangedBy.
Move()
Remove()

Notification object

Contains address and schedule information for sending pager or e-mail message notifications to the user. See "Notification objects" on page 2-16 for a detailed description of the Notification object structure.

Properties:

Address as .
ID as .
Level as .
MessageAttachment as .
Parent as .
ScheduleItems as .
Session as .
Tag as .
UseScheduling as .
NotificationType as .

NotifyScheduleItem object

Contains a record of when pager or e-mail notifications should be sent. NotifyScheduleItem objects can be collected in a NotifyScheduleItems object.

Properties:

Enabled as .
ID as .
Parent as .
CustomSchedule as .
ScheduleType as .
Tag as .

NotifyScheduleItems object

A collection of NotifyScheduleItem objects.

Methods:

Add() as **NotifyScheduleItem**

Discard()

Exists(Index, [SearchType]) as **Boolean**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) as **NotifyScheduleItem**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Remove(Index, [SearchType])

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Save()

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

Parties object

A collection of Party objects.

Methods:

Exists(Index, [SearchType]) as **Boolean**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) as **Party**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

Party object

Contains information about one of the parties involved in a call, such as the caller, the recipient, a member of a conference, and so forth. Party objects can be collected in a Parties object.

Methods:

Announce()

Answer()

Associate(Contact)

Contact as Contact.

Disconnect()

Hold([PlayHoldMusic])

PlayHoldMusic as Boolean. Optional. Default value is -1 (true).

SetCustomData(Key, Value)

Key as String.

Value as String.

SetRole(Role, [Partner])

Role as TVPartyRole.

Partner as Party. Optional. Default value is <unprintable IDispatch*>.

Supports(Feature) as Boolean

Feature as TVPartyFeature.

Properties:

Address as .

AnsweredTime as .

AssociatedPerson as .

AssociatedPersonID as .

AssociatedPersonType as .

AudioClip as .

Class as .

Comments as .

CustomData as .

Device as .

DID as .

Direction as .

Features as .

FirstName as .

ID as .

LastName as .

Mute as .

Name as .

Number as .

Parent as .

Role as .

Session as .

StartTime as .

Status as .

Tag as .

Handle as .

WaitStartTime as .

PartyHistories object

A collection of PartyHistory objects.

Methods:

Exists(Index, [SearchType]) **as Boolean**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) **as PartyHistory**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

PartyHistory object

Contains information concerning one party in a call. PartyHistory objects can be collected in a PartyHistories object.

Properties:

AccountCode as .

AnsweredBy as .

AnsweredTime as .

CallerIDAddress as .

CallerIDName as .

CallerIDNumber as .

CustomData as .

DeviceNumber as .

DialString as .

DID as .

Direction as .

EndTime as .

ID as .

LeftMessage as .

Message as .

Parent as .

Person as .

PersonID as .

Result as .

StartTime as .

Tag as .

OrganizationName as .

Permission object

Contains a record of a permission needed for a specific feature. Permission objects can be collected in a Permissions object.

Properties:

ID as .
Name as .
OwnerID as .
Parent as .
PermissionType as .
Session as .
Tag as .
UseDefault as .
Value as .

Permissions object

A collection of Permission objects.

Methods:

Add([Everyone], [User]) **as Permission**
Everyone as Boolean. Optional. Default value is 0 (false).
User as Item. Optional.

Discard()

Exists(Index, [SearchType]) **as Boolean**
Index as Variant.
SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) **as Permission**
Index as Variant.
SearchType as TVSearchType. Optional. Default value is 0.

Remove(Index, [SearchType])
Index as Variant.
SearchType as TVSearchType. Optional. Default value is 0.

Save()

Properties:

Count as .
NewEnum as .
Parent as .
Session as .
Tag as .

PersonalStatus object

Contains a record of a personal status, as displayed in the Client's Personal Status view. PersonalStatus objects can be collected in a folder of type tvFolderPersonalStatus (see "Folder types" on page 2-10). See "PersonalStatus objects" on page 2-16 for a detailed description of the PersonalStatus object structure.

Methods:

Delete([Mode])

Mode as TVDeleteMode. Optional. Default value is 0.

Discard()

Save([Overwrite])

Overwrite as Boolean. Optional.

Properties:

AcceptQueueCalls as .

CallForwarding as .

Class as .

Description as .

DisplayName as .

DoNotRingPhone as .

ForwardingAddress as .

ForwardingAttemptCentrexPBXTransfer as .

ForwardingPromptForPassword as .

ForwardingPromptToAccept as .

ForwardingRingDuration as .

Greeting as .

ID as .

Name as .

Parent as .

PromptForDescription as .

RoutingList as .

Session as .

Synchronized as .

System as .

Tag as .

UseCallRules as .

VoiceTitle as .

Events:

Change(Synchronized, ChangedBy)

Synchronized as Boolean.

ChangedBy as TVItemChangedBy.

Remove()

PhoneGatewayService object

A dialing service object containing information for dialing a number over an IP Gateway.

PhoneGatewayService objects can be collected in a folder of type tvFolderServices (see "Folder types" on page 2-10). See "Dialing Service objects" on page 2-14 for information about all of the dialing service objects.

Methods:

Delete([Mode])

Mode as TVDeleteMode. Optional. Default value is 0.

Discard()

Move(Destination)

Destination as Folder.

Save([Overwrite])

Overwrite as Boolean. Optional.

Properties:

AccessCode as .

AddressType as .

AreaCode as .

Class as .

CountryCode as .

Hidden as .

ID as .

IncludeLongDistancePrefix as .

InternationalPrefix as .

LongDistancePrefix as .

Name as .

Parent as .

Session as .

Synchronized as .

Tag as .

Events:

Change(Synchronized, ChangedBy)

Synchronized as Boolean.

ChangedBy as TVItemChangedBy.

Remove()

PhoneService object

A dialing service object containing information for dialing a number over a standard phone line. PhoneService objects can be collected in a folder of type tvFolderServices (see "Folder types" on page 2-10). See "Dialing Service objects" on page 2-14 for information about all of the dialing service objects.

Methods:

Delete([Mode])

Mode as TVDeleteMode. Optional. Default value is 0.

Discard()

Move(Destination)

Destination as Folder.

Save([Overwrite])

Overwrite as Boolean. Optional.

Properties:

AccessCode as .

AddressType as .

AreaCode as .

Class as .

CountryCode as .

Default as .

Hidden as .

ID as .

IncludeLongDistancePrefix as .

InternationalPrefix as .

LongDistancePrefix as .

Name as .

Parent as .

Session as .

Synchronized as .

Tag as .

Events:

Change(Synchronized, ChangedBy)

Synchronized as Boolean.

ChangedBy as TVItemChangedBy.

Remove()

QueueByPeriodStat object

Contains information concerning a Call Center queue's activity during a specific period of time.

Properties:

ItemsAbandonedTotalInbound as .
ItemsActiveTotalInbound as .
ItemsCompletedTotalInbound as .
ItemsReceivedTotalInbound as .
Parent as .
Tag as .
TalkTimeTotalInbound as .
WaitTimeTotalForAbandonedItemsInbound as .
WaitTimeTotalForAnsweredItemsInbound as .
ItemsCompletedTotalOutbound as .
ItemsConnectedTotalOutbound as .
ItemsPlacedTotalOutbound as .
LongestItemDurationOutbound as .
LongestTalkTimeOutbound as .
LongestWrapupTimeOutbound as .
NumberRedirectMaxHold as .
NumberRedirectQueueBusy as .
NumberRedirectQueueClosed as .
TalkTimeTotalOutbound as .
WrapupTimeTotalOutbound as .
LongestItemDurationInbound as .
LongestTalkTimeInbound as .

QueueByShiftStat object

Contains information concerning a Call Center queue's activity during a specific shift.

Properties:

ItemsAbandonedTotalInbound as .
ItemsActiveTotalInbound as .
ItemsCompletedTotalInbound as .
ItemsReceivedTotalInbound as .
Parent as .
Tag as .
TalkTimeTotalInbound as .
WaitTimeTotalForAbandonedItemsInbound as .
WaitTimeTotalForAnsweredItemsInbound as .
ItemsCompletedTotalOutbound as .
ItemsConnectedTotalOutbound as .
ItemsPlacedTotalOutbound as .
LongestItemDurationOutbound as .
LongestTalkTimeOutbound as .
LongestWrapupTimeOutbound as .
NumberRedirectMaxHold as .
NumberRedirectQueueBusy as .
NumberRedirectQueueClosed as .
TalkTimeTotalOutbound as .
WrapupTimeTotalOutbound as .
LongestItemDurationInbound as .
LongestTalkTimeInbound as .

QueueStat object

Contains information concerning a Call Center queue's activity. QueueStat objects can be collected in a folder of type tvFolderQueueStats (see "Folder types" on page 2-10).

Properties:

AgentsAvailable as .
AgentsSignedIn as .
AgentsWrapUpInbound as .
Class as .
CurrentPeriodStat as .
CurrentShiftStat as .
ID as .
ItemsAbandonedTotalInbound as .
ItemsActiveTotalInbound as .
ItemsCompletedTotalInbound as .
ItemsReceivedTotalInbound as .
ItemsWaitingTotalInbound as .
LongestItemDurationInbound as .
LongestTalkTimeInbound as .
LongestWaitTimeInbound as .
Parent as .
PreviousPeriodStat as .

PreviousShiftStat as .
QueueID as .
QueueName as .
QueueStatus as .
Session as .
StartTime as .
Synchronized as .
Tag as .
TalkTimeTotalInbound as .
WaitTimeTotalForAbandonedItemsInbound as .
WaitTimeTotalForAnsweredItemsInbound as .
ItemsBeingPlacedTotalOutbound as .
ItemsCompletedTotalOutbound as .
ItemsConnectedTotalOutbound as .
ItemsPlacedTotalOutbound as .
LongestItemDurationOutbound as .
LongestTalkTimeOutbound as .
LongestWrapupTimeOutbound as .
NumberRedirectMaxHold as .
NumberRedirectQueueBusy as .
NumberRedirectQueueClosed as .
TalkTimeTotalOutbound as .
WrapupTimeTotalOutbound as .

Events:

Change()
Remove()

Recipients object

Collection object containing a list of Contact and Workgroup items.

Methods:

Add(Recipient)

Recipient as *Item*.

Discard()

Exists(Index, [SearchType]) **as Boolean**

Index as *Variant*.

SearchType as *TVSearchType*. Optional. Default value is 0.

Item(Index, [SearchType]) **as IDispatch**

Index as *Variant*.

SearchType as *TVSearchType*. Optional. Default value is 0.

Remove(Index, [SearchType])

Index as *Variant*.

SearchType as *TVSearchType*. Optional. Default value is 0.

Save()

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

Role object

Contains information on a given set of permissions that apply to a group of Users. Any user added to a role will inherit permissions from that role.

Properties:

Class as .

Comments as .

Name as .

ID as .

Parent as .

Tag as .

Roles object

A collection of Role objects.

Methods:

Add(Role)

Role as *Role*.

Discard()

Exists(Index, [SearchType]) **as Boolean**

Index as *Variant*.

SearchType as *TVSearchType*. Optional. Default value is 0.

Item(Index, [SearchType]) **as Role**

Index as *Variant*.

SearchType as TVSearchType. Optional. Default value is 0.

Remove(Index, [SearchType])

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Save()

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

RoutingList object

Contains information about a routing list, similar to the information displayed in the Client's Routing Lists view. RoutingList objects can be collected in a folder of type tvFolderRoutingLists (see "Folder types" on page 2-10). See "RoutingList objects" on page 2-17 for a detailed description of the RoutingList object structure.

Methods:

Delete([Mode])

Mode as TVDeleteMode. Optional. Default value is 0.

Discard()

Save([Overwrite])

Overwrite as Boolean. Optional. Default value is 0 (false).

Properties:

Actions as .

Active as .

Class as .

Customizable as .

Default as .

FinalAction as .

ID as .

Name as .

Parent as .

Session as .

Synchronized as .

Tag as .

Events:

Change(Synchronized, ChangedBy)

Synchronized as Boolean.

ChangedBy as TVItemChangedBy.

Remove()

RoutingListAction object

Contains a record of a single routing list action. RoutingListAction objects can be collected in a RoutingListActions object.

Properties:

ActionType as .
Address as .
AttemptCentrexPBXTransfer as .
Description as .
Greeting as .
ID as .
Parent as .
PauseDuration as .
Position as .
PromptCallerBeforeAction as .
PromptCallerGreeting as .
PromptForPassword as .
PromptToAccept as .
RingDuration as .
Tag as .
WorkgroupRoutingType as .

RoutingListActions object

A collection of RoutingListAction objects.

Methods:

Add() as RoutingListAction
Discard()
Exists(Index, [SearchType]) as Boolean
Index as Variant.
SearchType as TVSearchType. Optional. Default value is 0.
Item(Index, [SearchType]) as RoutingListAction
Index as Variant.
SearchType as TVSearchType. Optional. Default value is 0.
Remove(Index, [SearchType])
Index as Variant.
SearchType as TVSearchType. Optional. Default value is 0.
Save()

Properties:

Count as .
NewEnum as .
Parent as .
Session as .
Tag as .

RoutingListFinalAction object

Contains a record of the final action in a routing list.

Properties:

ActionType as .
Address as .
Greeting as .
ID as .
Parent as .
PauseDuration as .
PlayGreeting as .
Tag as .

RoutingService object

Contains information for selecting a dialing service based on the number dialed.

Methods:

Delete([Mode])
Mode as TVDeleteMode. Optional. Default value is 0.
Discard()
Move(Destination)
Destination as Folder.
Save([Overwrite])
Overwrite as Boolean. Optional.

Properties:

AccessCode as .
AddressType as .
AreaCode as .
Class as .
CountryCode as .
Default as .
Hidden as .
ID as .
IncludeLongDistancePrefix as .
InternationalPrefix as .
LongDistancePrefix as .
Name as .
Parent as .
Session as .
Synchronized as .
Tag as .

Events:

Change(Synchronized, ChangedBy)
Synchronized as Boolean.
ChangedBy as TVItemChangedBy.
Remove()

Schedule object

Contains a record of the times when pager and e-mail message notifications should be sent. See “Schedule objects” on page 2-18 for a detailed description of the Schedule object structure. Schedule objects can be collected in a Schedules object.

Properties:

ID as .
Name as .
Parent as .
ScheduledDays as .
ScheduledDates as .
Session as .
Tag as .

ScheduledDate object

Contains a record of a date and time when pager and e-mail message notifications should be sent. ScheduledDate objects can be collected in a ScheduledDates object.

Properties:

EndTime as .
ID as .
Parent as .
Session as .
StartTime as .
Tag as .
Value as .

ScheduledDates object

A collection of ScheduledDate objects.

Methods:

Add(ScheduledDate) as **ScheduledDate**
ScheduledDate as Date.

Discard()

Exists(Index, [SearchType]) as **Boolean**
Index as Variant.
SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) as **ScheduledDate**
Index as Variant.
SearchType as TVSearchType. Optional. Default value is 0.

Remove(Index, [SearchType])
Index as Variant.
SearchType as TVSearchType. Optional. Default value is 0.

Save()

Properties:

Count as .
NewEnum as .

Parent as .

Session as .

Tag as .

ScheduledDay object

Contains a record of times on specific days of the week when pager and e-mail message notifications should be sent. ScheduledDay objects can be collected in a ScheduledDays object.

Properties:

EndTime as .

ID as .

Parent as .

Session as .

StartTime as .

Tag as .

Value as .

ScheduledDays object

A collection of ScheduledDay objects.

Methods:

Add(ScheduledDay) as **ScheduledDay**

ScheduledDay as TVScheduledDaysWeekday.

Discard()

Exists(Index, [SearchType]) as **Boolean**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) as **ScheduledDay**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Remove(Index, [SearchType])

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Save()

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

Schedules object

A collection of Schedule objects.

Methods:

Add(Name) as Schedule

Name as String.

Discard()**Exists(Index, [SearchType]) as Boolean**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) as Schedule

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Remove(Index, [SearchType])

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Save()

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

Session object

The root of the Client API. See "Structure of a session" on page 2-4 for details.

Methods:

CreateAddress([Value], [AddressType], [AddressSubType], [Service], [Category]) as Address

Value as String. Optional.

AddressType as TVAddressType. Optional. Default value is 1.

AddressSubType as TVAddressSubType. Optional.

Service as IDialingService. Optional.

Category as TVAddressCategory. Optional.

CreateCall(Address) as Call

Address as Address.

CreateCallAbort()**CreateItem(Class) as IDispatch**

Class as TVObjectClass.

CreateSession(Username, [ServerConnectionLevel]) as Session

Username as String.

ServerConnectionLevel as TVServerConnectionLevel. Optional. Default value is 0.

EmptyDeletedFolder([DeletedFolder], [Interval])

DeletedFolder as Folder. Optional.

Interval as Long. Optional. Default value is 0.

Export(FileName, source, [Parameters]) as Long

FileName as String.

source as Folder.

Parameters as Variant. Optional.

ExportCancel()**ExportDebugInfo(DataMember, [Password])**

DataMember as Long.

Password as String. Optional.

Find(Filter, ItemClass) as IDispatch

Filter as String.

ItemClass as TVObjectItemClass.

FindNext() as IDispatch**GetCallingInfo([LoggedInUserID], [LoggedInUserFirstName], [LoggedInUserLastName], [CallingAsID], [CallingAsFirstName], [CallingAsLastName])**

LoggedInUserID as String. Optional. Default value is "".

LoggedInUserFirstName as String. Optional. Default value is "".

LoggedInUserLastName as String. Optional. Default value is "".

CallingAsID as String. Optional. Default value is "".

CallingAsFirstName as String. Optional. Default value is "".

CallingAsLastName as String. Optional. Default value is "".

GetData(DataMember, [PreserveState], [LockType], [Password], [Filter]) as Recordset

DataMember as Long.

PreserveState as Boolean. Optional. Default value is -1 (true).

LockType as LockTypeEnum. Optional. Default value is 1.

Password as String. Optional. Default value is "".

Filter as String. Optional. Default value is "".

GetDefaultFolder(FolderType) as Folder

FolderType as TVDefaultFolders.

GetFolder(ID) as Folder

ID as String.

GetInterface(Interface, Item) as IDispatch

Interface as TVInterface.

Item as IDispatch.

GetItem(ID, [ParentID]) as IDispatch

ID as String.

ParentID as String. Optional. Default value is "".

GetItemFromTargetID(ID) as IDispatch

ID as String.

GetLastError() as Error**GetLastInboundCallAddress() as Address****GetPermissionValue(Name, [Parameters]) as Long**

Name as String.

Parameters as Variant. Optional.

GetRecentOutboundCallAddresses() as Collection**GetTarget(ID) as SystemTarget**

ID as String.

Import(FileName, Destination, Options, Mapping, HeaderPosition, [Parameters]) as Variant

FileName as String.

Destination as Folder.

Options as TVImportOptions.

Mapping as Variant.

HeaderPosition as TVImportHeaderPosition.

Parameters as Variant. Optional.

ImportCancel()

LogOff()

Logon(Server, Username, Password, Station, ApplicationType, ApplicationID, [ServerConnectionLevel], [CheckVersions], [StationUsage])

Server as String.

Username as String.

Password as String.

Station as Long.

ApplicationType as TVApplicationType.

ApplicationID as Long.

ServerConnectionLevel as TVServerConnectionLevel. Optional. Default value is 0.

CheckVersions as Boolean. Optional. Default value is -1 (true).

StationUsage as TVStationUsage. Optional. Default value is 0.

ReadImportRecord(FileName, RecordNumber) **as Collection**

FileName as String.

RecordNumber as Long.

StartRing(RingPattern, [NumberofRings])

RingPattern as TVUserRingPattern.

NumberofRings as Long. Optional. Default value is 2.

StopRing()

ValidateLogonInfo(Server, Username, Password, Station, [DefaultStation], [DefaultUsername]) **as TVSessionValidateLogonResult**

Server as String.

Username as String.

Password as String.

Station as Long.

DefaultStation as Long. Optional.

DefaultUsername as String. Optional.

ExportXML(FolderType, [ExportType]) **as String**

FolderType as TVDefaultFolders.

ExportType as TVExportType. Optional. Default value is 0.

ChangePassword(Server, Username, OldPassword, NewPassword, [MinimumPasswordLength])

Server as String.

Username as String.

OldPassword as String.

NewPassword as String.

MinimumPasswordLength as Long. Optional.

ChangeLocale(LocaleID)

LocaleID as String.

GetSystemCallHistory(Item) **as CallHistory**

Item as Variant.

GetLocale() **as String**

Properties:

ApplicationID as .

ApplicationType as .

Class as .

ClientServerTimeDiff as .

DefaultStationNumber as .

ExportRecords as .
Folders as .
ImportBytes as .
Locale as .
NameFormat as .
ShortcutGroups as .
Station as .
StationNumber as .
Status as .
System as .
SystemSettings as .
Tag as .
User as .
UserID as .
Version as .
AgentStatisticInterval as .

Events:

CallStatusChange(ID, Status, OldStatus)
ID as String.
Status as TVPartyStatus.
OldStatus as TVPartyStatus.

CreateCallAbort(ID)
ID as String.

CreateCallComplete(ID)
ID as String.

CreateCallStart(ID)
ID as String.

CallingInfoChange()

FolderAdd(ID, ParentID, Folder)
ID as String.
ParentID as String.
Folder as Folder.

FolderChange(ID, ParentID, Folder)
ID as String.
ParentID as String.
Folder as Folder.

FolderAfterMove(ID, ParentID, PreviousParentID)
ID as String.
ParentID as String.
PreviousParentID as String.

FolderAfterRemove(ID, ParentID)
ID as String.
ParentID as String.

FolderBeforeMove(ID, ParentID, Folder, Destination)
ID as String.
ParentID as String.
Folder as Folder.
Destination as Folder.

FolderBeforeRemove(ID, ParentID, Folder)

ID as String.

ParentID as String.

Folder as Folder.

ItemAdd(ID, Class, ParentID, Item)

ID as String.

Class as TVObjectClass.

ParentID as String.

Item as IDispatch.

ItemAfterMove(ID, Class, ParentID, PreviousParentID)

ID as String.

Class as TVObjectClass.

ParentID as String.

PreviousParentID as String.

ItemAfterRemove(ID, Class, ParentID)

ID as String.

Class as TVObjectClass.

ParentID as String.

ItemBeforeMove(ID, Class, ParentID, Item, Destination)

ID as String.

Class as TVObjectClass.

ParentID as String.

Item as IDispatch.

Destination as Folder.

ItemBeforeRemove(ID, Class, ParentID, Item)

ID as String.

Class as TVObjectClass.

ParentID as String.

Item as IDispatch.

ItemChange(ID, Class, ParentID, Item)

ID as String.

Class as TVObjectClass.

ParentID as String.

Item as IDispatch.

ItemDataExpired(ItemClass)

ItemClass as TVObjectItemClass.

PermissionChange()

SettingChange(Name, Value)

Name as String.

Value as Variant.

Logon(User)

User as User.

LogOff()

ServerShutdownScheduled(ShutdownDelay)

ShutdownDelay as Long.

ServerStatusChange(Status, ExtraInfo)

Status as TVServerStatus.

ExtraInfo as Variant.

Error(Error, SeverityOverride, Cancel)

Error as Error.

SeverityOverride as TVDebugSeverity.

Cancel as Boolean.

ShortcutGroupAdd(ID)

ID as String.

ShortcutGroupChange(ID)

ID as String.

ShortcutGroupRemove(ID)

ID as String.

ShortcutAdd(ID, GroupID)

ID as String.

GroupID as String.

ShortcutChange(ID, GroupID)

ID as String.

GroupID as String.

ShortcutRemove(ID, GroupID)

ID as String.

GroupID as String.

Shortcut object

Contains information that can be used to locate and access a specific Folder object. Shortcut objects can be collected in a Shortcuts object.

Properties:

FolderID as .

ID as .

ItemClass as .

Name as .

Parent as .

PictureID as .

Position as .

ResourceID as .

Session as .

Tag as .

ShortcutGroup object

Contains information about a group of shortcuts, for example, the General and Advanced shortcut groups displayed in the Client's view bar. ShortcutGroup objects can be collected in a ShortcutGroups object.

Methods:

Delete()
Discard()
Save()

Properties:

IconType as .
ID as .
Name as .
Parent as .
ResourceID as .
Session as .
Shortcuts as .
Tag as .

ShortcutGroups object

A collection of ShortcutGroup objects.

Methods:

Add([Name]) as ShortcutGroup
Name as String. Optional. Default value is "".

Exists(Index, [SearchType]) as Boolean
Index as Variant.
SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) as ShortcutGroup
Index as Variant.
SearchType as TVSearchType. Optional. Default value is 0.

Remove(Index, [SearchType])
Index as Variant.
SearchType as TVSearchType. Optional. Default value is 0.

Properties:

Count as .
NewEnum as .
Parent as .
Session as .
Tag as .

Shortcuts object

A collection of Shortcut objects.

Methods:

Add([Name], [FolderID]) **as Shortcut**

Name as String. Optional. Default value is "".

FolderID as String. Optional. Default value is "".

Discard()

Exists(Index, [SearchType]) **as Boolean**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) **as Shortcut**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Remove(Index, [SearchType])

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Save()

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

Station object

Contains configuration information about a station. See "Station objects" on page 2-19 for a detailed description of the Station object structure.

Methods:

Discard()

GetButtonCount() **as Long**

Save([Overwrite])

Overwrite as Boolean. Optional. Default value is 0 (false).

ValidateButtons(PDNButtonCount, SDNStations, SDNButtonCounts) **as Boolean**

PDNButtonCount as Long.

SDNStations as Long.

SDNButtonCounts as Long.

Properties:

Buttons as .

CadenceID as .

Class as .

DefaultIdleString as .

DefaultVolume as .

DeviceName as .

HandsFreeCanPlaceCall as .

HandsFreeDialtoneTimeout as .

HandsFreeDisconnectTimeout as .

HandsFreeModeEnabled as .
ID as .
Number as .
PadCharacter as .
PadExtension as .
Parent as .
PhoneFeatures as .
Session as .
Synchronized as .
Tag as .
TerminalType as .
ToneID as .
TransferMode as .
VoiceFirstAnsweringEnabled as .
DirectConnectDialString as .

Events:

Change(Synchronized, ChangedBy)
 Synchronized as Boolean.
 ChangedBy as TVItemChangedBy.
Remove()

StationButton object

Contains information about a specialized button on a feature phone. StationButton objects can be collected in a StationButtons object.

Properties:

Address as .
ButtonNumber as .
DirectoryStationNumber as .
Feature as .
Parent as .
RingDelay as .
Session as .
Tag as .

StationButtons object

A collection of StationButton objects.

Methods:

Add(ButtonNumber) as **StationButton**
 ButtonNumber as Long.
Discard()
Exists(Index, [SearchType]) as **Boolean**
 Index as Variant.
 SearchType as TVSearchType. Optional. Default value is 0.
Item(Index, [SearchType]) as **StationButton**
 Index as Variant.
 SearchType as TVSearchType. Optional. Default value is 0.

Remove(Index, [SearchType])

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Save()

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

StationFeature object

Contains information about a feature assigned to a StationButton object. StationFeature objects can be collected in a StationFeatures object. See "StationFeature objects" on page 2-19 for a detailed description of the StationFeature object structure.

Properties:

DisplayName as .

ID as .

Name as .

Parameters as .

Parent as .

ResourceID as .

Session as .

Tag as .

StationFeatures object

A collection of StationFeature objects.

Methods:

Exists(Index, [SearchType]) as **Boolean**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) as **StationFeature**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

StationParameter object

Contains a parameter required by a StationFeature object. StationParameter objects can be collected in a StationParameters object.

Properties:

ParameterType as .

Parent as .

Session as .

Tag as .

StationParameters object

A collection of StationParameter objects.

Methods:

Exists(Index, [SearchType]) as Boolean

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) as StationParameter

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

StationType object

Contains a list of features used by a specific type of feature phone. StationType objects can be collected in a StationTypes object.

Properties:

Features as .

Name as .

Parent as .

Session as .

Tag as .

StationTypes object

A collection of StationType objects.

Methods:

Exists(Index, [SearchType]) as **Boolean**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) as **StationType**

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

SwitchGatewayService object

A dialing service object containing information for dialing a number over a Switch Gateway.

SwitchGatewayService objects can be collected in a folder of type tvFolderServices (see "Folder types" on page 2-10). See "Dialing Service objects" on page 2-14 for information about all of the dialing service objects.

Methods:

Delete([Mode])

Mode as TVDeleteMode. Optional. Default value is 0.

Discard()

Move(Destination)

Destination as Folder.

Save([Overwrite])

Overwrite as Boolean. Optional.

Properties:

AccessCode as .

AddressType as .

Class as .

Hidden as .

ID as .

Name as .

Parent as .

Session as .

Synchronized as .

Tag as .

Events:

Change(Synchronized, ChangedBy)

Synchronized as Boolean.

ChangedBy as TVItemChangedBy.

Remove()

SwitchService object

A dialing service object containing information for dialing a number over a Switch. SwitchService objects can be collected in a folder of type tvFolderServices (see "Folder types" on page 2-10). See "Dialing Service objects" on page 2-14 for information about all of the dialing service objects.

Methods:

Delete([Mode])

Mode as TVDeleteMode. Optional. Default value is 0.

Discard()

Move(Destination)

Destination as Folder.

Save([Overwrite])

Overwrite as Boolean. Optional.

Properties:

AccessCode as .

AddressType as .

Class as .

Hidden as .

ID as .

Name as .

Parent as .

Session as .

Synchronized as .

Tag as .

Events:

Change(Synchronized, ChangedBy)

Synchronized as Boolean.

ChangedBy as TVItemChangedBy.

Remove()

System object

Represents the system as a whole. It serves as the main interface between the Client and the Server. It also contains all system-wide settings and licenses. It handles all operations that affect the system as a whole. For example, the System object provides methods to backup or restore the database, and to start or shut down the Server. See "The System object" on page 2-5 for a detailed description of the System object structure.

Methods:

GetDeviceName(DeviceNumber) as String

DeviceNumber as Long.

GetLicenseStatus(ExpirationDate) as TVLicenseStatus

ExpirationDate as Date.

Properties:

BusinessSchedules as .

Class as .

DefaultInternetService as .

DefaultLocaleCode as .

DefaultPhoneService as .
LocaleCodes as .
MinPasswordLength as .
ServerName as .
ServerStatus as .
ServerTime as .
ServerTimeZoneBias as .
Session as .
StationTypes as .
Tag as .
TimeUntilServerShutDown as .
VARCompany as .
VARContactInfo as .
VARName as .
DefaultOutboundCallerIDPresentation as .
DefaultOutboundCallerIDName as .
DefaultOutboundCallerIDNumber as .
Roles as .

SystemSetting object

Contains information about a system setting. SystemSetting objects can be collected in a SystemSettings object.

Methods:

Reset()
Save()

Properties:

AllowChange as .
Class as .
Customizable as .
DataType as .
Default as .
Description as .
ID as .
Location as .
Name as .
Session as .
Tag as .
Value as .

SystemSettings object

A collection of SystemSetting objects.

Methods:

Exists(Index, [SearchType]) as Boolean

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

GetValue(Name) as Variant

Name as String.

GetValueEx(Name, DefaultValue) as Variant

Name as String.

DefaultValue as Variant.

Item(Index, [SearchType]) as SystemSetting

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

ResetValue(Name)

Name as String.

SetValue(Name, Value)

Name as String.

Value as Variant.

Properties:

Count as .

NewEnum as .

Session as .

Tag as .

SystemTarget object

Contains information about a user, similar to the information displayed in the Client's Extensions view. SystemTarget objects can be collected in a folder of type tvFolderSystemTarget (see "Folder types" on page 2-10).

Methods:

ApplyStatus(StatusType, [Password])

StatusType as TVPersonalStatusType.

Password as String. Optional. Default value is "".

Properties:

Agent as .

Availability as .

CallForwardingType as .

CallForwardingSystemTarget as .

Class as .

Comments as .

DIDList as .

Extension as .

FirstName as .

HasMailbox as .

ID as .

JobTitle as .

LastName as .
Name as .
Parent as .
QueueStandby as .
Session as .
StatusDescription as .
StatusName as .
Synchronized as .
Tag as .
TargetType as .

Events:

Change(Synchronized)
Synchronized as Boolean.
Remove()

User object

A Session's User object represents the end user who is currently logged in. User objects can be collected in a folder of type tvFolderUsers (see "Folder types" on page 2-10). See "The User object" on page 2-6 for a detailed description of the User object structure.

Methods:

ApplyStatus(Status)
Status as PersonalStatus.
ChangePassword(NewPassword, [OldPassword])
NewPassword as String.
OldPassword as String. Optional. Default value is "".
Delete([Mode])
Mode as TVDeleteMode. Optional. Default value is 0.
Discard()
Save([Overwrite])
Overwrite as Boolean. Optional.

Properties:

ACDDoNotDisturb as .
AcceptQueueCalls as .
Addresses as .
Agents as .
AutoAccountCodeLookup as .
AnnounceCallBehavior as .
BusinessSchedule as .
CallBackEnabled as .
CallBackInterval as .
CallHistoryDayLimit as .
CallRecordingRecipient as .
CallWaiting as .
Category as .
CentrexPBXTransferOnForward as .
Class as .
Comments as .

Company as .
DefaultAddress as .
DefaultAudioDevice as .
DefaultAudioSkipInterval as .
DefaultCentrexPBXTransferOnForward as .
DefaultContactAction as .
DefaultForwardingAddress as .
DefaultForwardRingDuration as .
DefaultPromptForwarderForPassword as .
DefaultPromptForwarderToAccept as .
DialByNameAction as .
DIDList as .
DoNotRingPhone as .
EmailNotification as .
EmptyDeletedBehavior as .
EmptyDeletedInterval as .
ExchangeMailbox as .
ExchangeSynchronization as .
Extension as .
ExternalCallRingPattern as .
ExternalCallTypes as .
FirstName as .
ForwardingAddress as .
ForwardRingDuration as .
FullName as .
GreetingAndTitleSize as .
HoldAudio as .
ID as .
InboundAccountCodeBehavior as .
InboundCallBehavior as .
InternalCallRingPattern as .
JobTitle as .
LastAppliedStatus as .
LastName as .
MailboxSize as .
MaxConferenceMembers as .
MaxGreetingAndTitleSize as .
MaxMailboxSize as .
MaxMessageDuration as .
MessagePlayBackOrder as .
NameFormat as .
Operator as .
OutboundAccountCodeBehavior as .
OutboundCallBehavior as .
PagerNotification as .
Parent as .
Permissions as .
PlayNewMessageSound as .
PromptCallerForName as .

PromptForStatusDescription as .
PromptForStatusDuration as .
PromptForwarderForPassword as .
PromptForwarderToAccept as .
RingDuration as .
Session as .
SendDigitsBehavior as .
Station as .
StationNumber as .
StatusDescription as .
Synchronized as .
Tag as .
TelephonePromptLanguage as .
UseCallRules as .
VerifyIPAddress as .
VerifyPhoneAddress as .
VoiceTitle as .
AllowVoiceMailLogonDuringGreeting as .
CallingAsFirstName as .
CallingAsID as .
CallingAsLastName as .
CallNotification as .
LogCalls as .
OutboundCallerIDNumber as .
OutboundCallerIDPresentation as .
Roles as .

Events:

Change(Synchronized, ChangedBy)
 Synchronized as Boolean.
 ChangedBy as TVItemChangedBy.
Remove()

View object

Contain informations about a Client view. View objects can be collected in a Views object.

Methods:

Export([ExportType]) as **String**
 ExportType as TVExportType. Optional. Default value is 0.
Filter([FieldName], [Operator], [Value])
 FieldName as String. Optional. Default value is "".
 Operator as TVFilterOperator. Optional. Default value is 0.
 Value as Variant. Optional.
GetData([PreserveState], [Password], [ShowAll], [ResetDataSource]) as **Recordset**
 PreserveState as Boolean. Optional. Default value is -1 (true).
 Password as String. Optional.
 ShowAll as Boolean. Optional.
 ResetDataSource as Boolean. Optional.
Reset()
Sort([FieldName], [SortOrder])

FieldName as String. Optional. Default value is "".

SortOrder as TVSortOrder. Optional. Default value is 1.

ExportItemClass([ExportType]) as String

ExportType as TVExportType. Optional. Default value is 0.

Properties:

Columns as .

Current as .

DisplayName as .

FilterString as .

ID as .

Name as .

Parent as .

ResourceID as .

Session as .

SortString as .

SystemDefined as .

Tag as .

Views object

A collection of View objects. See "The Folder object" on page 2-8 for details about Folder object structures.

Methods:

Exists(Index, [SearchType]) as Boolean

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Item(Index, [SearchType]) as View

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

Workgroup object

Contains information about a workgroup, similar to the information displayed in the Client's Workgroups view. Workgroup objects can be collected in a WorkgroupMembers object or in a folder of type tvFolderWorkgroups (see "Folder types" on page 2-10). See "Workgroup objects" on page 2-20 for a detailed description of the Workgroup object structure.

Methods:

Delete([Mode])

Mode as TVDeleteMode. Optional. Default value is 0.

Discard()

Save([Overwrite])

Overwrite as Boolean. Optional. Default value is 0 (false).

Properties:

Class as .
Comments as .
DialByNameAction as .
DIDList as .
Extension as .
ID as .
Members as .
Name as .
Operator as .
Parent as .
RingDuration as .
Session as .
Synchronized as .
Tag as .
VoiceTitle as .

Events:

Change(Synchronized, ChangedBy)
 Synchronized as Boolean.
 ChangedBy as TVItemChangedBy.
Remove()

WorkgroupMember object

Contains either a Contact object or a Workgroup object. WorkgroupMember objects can be collected in a WorkgroupMembers object.

Properties:

Member as .
ID as .
Parent as .
Position as .
Session as .
Tag as .

WorkgroupMembers object

A collection of WorkgroupMember objects.

Methods:

Add([Member]) as **WorkgroupMember**
 Member as IItem. Optional. Default value is <unprintable IDispatch*>.
Discard()
Exists(Index, [SearchType]) as **Boolean**
 Index as Variant.
 SearchType as TVSearchType. Optional. Default value is 0.
Item(Index, [SearchType]) as **WorkgroupMember**
 Index as Variant.
 SearchType as TVSearchType. Optional. Default value is 0.
Remove(Index, [SearchType])

Index as Variant.

SearchType as TVSearchType. Optional. Default value is 0.

Save()

Properties:

Count as .

NewEnum as .

Parent as .

Session as .

Tag as .

CLIENT API ENUMERATIONS

This appendix lists the TeleVantage Client API enumerations and their values. For more information about the Client API, see Chapter 2. For a detailed object reference, see Appendix D.

TVAddressCategory

tvAddressCategoryNone = 0
tvAddressCategoryBusiness = 1
tvAddressCategoryHome = 2
tvAddressCategoryMobile = 3
tvAddressCategoryBusiness2 = 4
tvAddressCategoryHome2 = 5
tvAddressCategoryOther = 6

TVAddressError

tvAddressErrorNone = 0
tvAddressErrorInvalidParent = 1851
tvAddressErrorSetCallerIDParent = 1852
tvAddressErrorCallerIDConflict = 1853
tvAddressErrorDefaultToNothing = 1854
tvAddressErrorValueEmpty = 1855
tvAddressErrorCallerIDUsage = 1856
tvAddressErrorDuplicateCategory = 1857
tvAddressErrorInvalidService = 1858
tvAddressErrorInvalidType = 1859
tvAddressErrorUnableToRemoveCallerID = 1860
tvAddressErrorUnableToSetDefault = 1861
tvAddressErrorInvalidCategory = 1862
tvAddressErrorCouldNotConvertToPhone = 1863
tvAddressErrorAmbiguous = 1864
tvAddressErrorInvalidStation = 1865
tvAddressErrorInvalidID = 1866
tvAddressErrorInvalidUseRule = 1867
tvAddressErrorDefaultForUser = 1868

TVAddressFieldValidation

tvAddressFieldValidationMaxDescription = 5000
tvAddressFieldValidationMaxValue = 255
tvAddressFieldValidationMaxAccountCode = 255

TVAddressPhoneComponent

tvAddressPhoneComponentAreaCode = 0
tvAddressPhoneComponentCountryCode = 1
tvAddressPhoneComponentPhoneNumber = 2
tvAddressPhoneComponentUseRules = 3

TVAddressSubType

tvAddressSubTypeDoNotUseRules = 0
tvAddressSubTypeUseRules = 8
tvAddressSubTypePager = 16

TVAddressType

tvAddressTypePhoneNumber = 1
tvAddressTypeIP = 2
tvAddressTypeStation = 3
tvAddressTypeUser = 4
tvAddressTypeEmail = 5
tvAddressTypeCustom = 6
tvAddressTypeExtension = 6
tvAddressTypeName = 7
tvAddressTypeApplication = 8
tvAddressTypePlugin = 8
tvAddressTypeAutoAttendant = 9
tvAddressTypeWorkgroup = 10
tvAddressTypeQueue = 11
tvAddressTypeUnused = 12
tvAddressTypeDID = 12
tvAddressTypeUserDefaultLocation = 13
tvAddressTypeVoiceMail = 14
tvAddressTypeVoiceMailAccess = 15
tvAddressTypeAccountLogon = 15
tvAddressTypeHoldMenu = 16
tvAddressTypeUnparkMenu = 21

TVAddressUsageType

tvAddressUsageAddress = 1
tvAddressUsageCallerID = 2
tvAddressUsageBoth = 3

TVAgentStatisticInterval

tvAgentStatisticIntervalDay = 0
tvAgentStatisticIntervalShift = 1

TVAgentStatus

tvAgentStatusSignedOut = 0
tvAgentStatusAvailable = 1
tvAgentStatusStandby = 2
tvAgentStatusWrapUp = 3
tvAgentStatusOffering = 4
tvAgentStatusNoAnswer = 5
tvAgentStatusActiveInbound = 6
tvAgentStatusActiveOutbound = 7
tvAgentStatusWrapUpInbound = 3
tvAgentStatusWrapUpOutbound = 8
tvAgentStatusDialing = 9

TVApplicationType

tvApplicationTypeClient = 0
tvApplicationTypeAdministrator = 1

TVAssociatedPersonType

tvAssociatedPersonTypeSystem = 0
tvAssociatedPersonTypeUser = 1
tvAssociatedPersonTypeContact = 2
tvAssociatedPersonTypeUnknown = 3
tvAssociatedPersonTypeQueue = 4

TVAudioDeviceType

tvAudioDevicePhone = 0
tvAudioDeviceSpeakers = 1

TVAudioState

tvAudioStatePlaying = 1
tvAudioStateRecording = 2
tvAudioStatePausedPlaying = 3
tvAudioStatePausedRecording = 4
tvAudioStateIdle = 5
tvAudioStateUnknown = 6
tvAudioStateRinging = 7

TVAudioStateChangeReason

tvAudioStateChangeReasonPlay = 0
tvAudioStateChangeReasonPause = 1
tvAudioStateChangeReasonRecord = 2
tvAudioStateChangeReasonStop = 3
tvAudioStateChangeReasonSilentRange = 4
tvAudioStateChangeReasonDeviceChange = 5
tvAudioStateChangeReasonPlayDone = 6
tvAudioStateChangeReasonPlayInterrupt = 7
tvAudioStateChangeReasonSessionDone = 8
tvAudioStateChangeReasonSessionRinging = 9
tvAudioStateChangeReasonRecordTerminateUser = 10
tvAudioStateChangeReasonRecordTerminateMaxSilence = 11
tvAudioStateChangeReasonRecordTerminateTimeOut = 12
tvAudioStateChangeReasonRecordTerminateDigit = 13

TVAudioType

tvAudioTypeMessage = 2
tvAudioTypeGreeting = 1
tvAudioTypeVoiceTitle = 0

TVAudioVOXFormat

tvAudioVOXFormatServerDefault = 0xffffffff
tvAudioVOXFormatALaw = 0
tvAudioVOXFormatULaw = 1

TVCallAnnounceType

tvCallAnnounceTypeCallee = 0
tvCallAnnounceTypeCaller = 1

TVCallDeclineMode

tvCallDeclineToRoutingList = 0
tvCallDeclineToVoiceMail = 1
tvCallDeclineToVoiceMailScreen = 2

TVCallDirection

tvCallDirectionInbound = 0
tvCallDirectionOutbound = 1

TVCallerIDFormat

tvCallerIDFormatNameAndNumber = 0
tvCallerIDFormatName = 1
tvCallerIDFormatNumber = 2

TVCallFeature

tvCallFeatureAnnounce = 0
tvCallFeatureAnswer = 1
tvCallFeatureAssociate = 2
tvCallFeatureConference = 3
tvCallFeatureDecline = 4
tvCallFeatureDeclineToVoiceMail = 5
tvCallFeatureDeclineToVoiceMailScreen = 6
tvCallFeatureDisconnect = 7
tvCallFeatureHold = 8
tvCallFeatureJoin = 9
tvCallFeatureMute = 10
tvCallFeaturePark = 11
tvCallFeatureRecordPause = 12
tvCallFeatureRecordResume = 13
tvCallFeatureRecordStart = 14
tvCallFeatureRecordStop = 15
tvCallFeatureRoleCoach = 16
tvCallFeatureRoleMonitor = 17
tvCallFeatureRolePeer = 18
tvCallFeatureRolePupil = 19
tvCallFeatureScreenMessage = 20
tvCallFeatureSetAccountCode = 21
tvCallFeatureSetRole = 22
tvCallFeatureTransferBlind = 23
tvCallFeatureTransferBlindToApplication = 24
tvCallFeatureTransferSupervisedAbort = 25
tvCallFeatureTransferSupervisedComplete = 26
tvCallFeatureTransferSupervisedConference = 27
tvCallFeatureTransferSupervisedStart = 28
tvCallFeatureUnpark = 29

TVCallFieldValidation

tvCallFieldValidationMaxAccountCode = 255
tvCallFieldValidationMaxComments = 5000
tvCallFieldValidationMaxCustomData = 5000

TVCallGrabAndHoldAudioType

tvCallGrabAndHoldAudioTypeNone = 0
tvCallGrabAndHoldAudioTypeSystemDefault = 1
tvCallGrabAndHoldAudioTypeUserDefault = 2
tvCallGrabAndHoldAudioTypeCustom = 3

TVCallHistoryError

tvCallHistoryErrorUnsavedContact = 5501

TVCallHistoryFieldValidation

tvCallHistoryFieldValidationMaxNotes = 5000

tvCallHistoryFieldValidationMaxAccountCode = 255

TVCallHistoryResult

tvCallHistoryResultUnknown = 0xffffffff

tvCallHistoryResultAbandoned = 0

tvCallHistoryResultConnected = 1

tvCallHistoryResultLeftMessage = 2

tvCallHistoryResultBlindTransfer = 3

tvCallHistoryResultSupervisedTransfer = 4

tvCallHistoryResultLogin = 5

tvCallHistoryResultNoAnswer = 6

tvCallHistoryResultLoginFailed = 7

TVCallRecordFormat

tvCallRecordFormatADPCM6K = 0

tvCallRecordFormatADPCM8K = 1

tvCallRecordFormatPCM6K = 2

tvCallRecordFormatPCM8K = 3

TVCallRecordStatus

tvCallRecordStatusIdle = 0

tvCallRecordStatusRecording = 1

tvCallRecordStatusPaused = 2

TVCallRuleCallerTypeCondition

tvCallRuleCallerTypeNone = 0

tvCallRuleCallerTypeInternal = 1

tvCallRuleCallerTypeExternal = 2

tvCallRuleCallerTypeUnidentified = 4

tvCallRuleCallerTypeSpecific = 8

TVCallRuleError

tvCallRuleErrorNonUniqueName = 7001
tvCallRuleErrorNoName = 7002
tvCallRuleErrorNonUniquePriority = 7003
tvCallRuleErrorNoPriority = 7004
tvCallRuleErrorInvalidCallerCondition = 7005
tvCallRuleErrorScheduleConditionInvalidCombo = 7006
tvCallRuleErrorCallerConditionInvalidCombo = 7007
tvCallRuleErrorNoConditions = 7008
tvCallRuleErrorNoActions = 7009
tvCallRuleErrorCannotSwapWithDifferentFolder = 7010

TVCallRuleFieldValidation

tvCallRuleFieldValidationMaxName = 255
tvCallRuleFieldValidationMinName = 1
tvCallRuleFieldValidationMinPriority = 1

TVCallRuleRingOverride

tvCallRuleRingDoNotOverride = 0
tvCallRuleRingAlways = 1
tvCallRuleRingNever = 2

TVCallRuleScheduleType

tvCallRuleScheduleTypeNone = 0
tvCallRuleScheduleTypeBusinessHours = 1
tvCallRuleScheduleTypeNonBusinessHours = 2
tvCallRuleScheduleTypeAfterHoursWorkDays = 3
tvCallRuleScheduleTypeNonWorkDays = 4
tvCallRuleScheduleTypeHolidays = 5
tvCallRuleScheduleTypeCustomHours = 6

TVCallType

tvCallTypeConsultation = 3
tvCallTypeNormal = 2
tvCallTypeParked = 5
tvCallTypeTransfer = 4

TVColumnsError

tvColumnsErrorNonUniqueName = 9626
tvColumnsErrorInvalidName = 9627

TVContactAssociateFlags

tvContactAssociateNoCallerID = 0
tvContactAssociateCallerIDNumberToCallerID = 1
tvContactAssociateCallerIDNumberToCategory = 2
tvContactAssociateCallerIDNameToCallerID = 4
tvContactAssociateCallbackNumberToCallerID = 8
tvContactAssociateCallbackNumberToCategory = 16
tvContactAssociateCategoryOther = 0
tvContactAssociateCategoryBusiness = 128
tvContactAssociateCategoryHome = 256
tvContactAssociateCategoryMobile = 512
tvContactAssociateCategoryBusiness2 = 1024
tvContactAssociateCategoryHome2 = 2048
tvContactAssociateNoVoiceTitle = 0
tvContactAssociateVoiceTitle = 16384

TVContactError

tvContactErrorDuplicatePIN = 1002
tvContactErrorMissingName = 1003
tvContactErrorCallerIDConflict = 1004
tvContactErrorTelephonePromptLanguage = 1005
tvContactErrorCannotClearDefault = 1006
tvContactErrorNotInAddresses = 1007
tvContactErrorNoPINInSharedFolder = 1008

TVContactFieldValidation

tvContactFieldValidationMaxComments = 5000
tvContactFieldValidationMaxCompanyName = 50
tvContactFieldValidationMaxFirstName = 64
tvContactFieldValidationMaxJobTitle = 30
tvContactFieldValidationMaxLastName = 30
tvContactFieldValidationMaxPIN = 6
tvContactFieldValidationMaxAccountCode = 255

TVDefaultFolders

tvFolderAgents = 100
tvFolderAgentStats = 2300
tvFolderCallHistory = 400
tvFolderCallRules = 1000
tvFolderCalls = 300
tvFolderContacts = 500
tvFolderCurrentUser = 0xffffffff
tvFolderDeleted = 0
tvFolderGreetings = 1500
tvFolderMessages = 700
tvFolderPersonalStatus = 1600
tvFolderQueueStats = 2200
tvFolderRoutingLists = 1800
tvFolderServices = 1100
tvFolderSystemTarget = 1700
tvFolderUsers = 1300
tvFolderWorkgroups = 2100

TVDeleteMode

tvDeleteMoveToDeleted = 0
tvDeletePermanent = 1
tvDeletePermanentPromptIfInUse = 2

TVDeviceHookState

tvDeviceHookStateOff = 1
tvDeviceHookStateOn = 0
tvDeviceHookStateUnknown = 2

TVDirectorySearchMode

tvDirectorySearchLastName = 0
tvDirectorySearchFirstName = 1
tvDirectorySearchEither = 2

TVError

tvErrorUnexpected = 1001
tvErrorDeleteWhileEdit = 7501
tvErrorMoveInvalidDestination = 7502
tvErrorMoveWhileNew = 7503
tvErrorMoveWhileEdit = 7504
tvErrorSaveOutOfSync = 7505
tvErrorObjectIsDeleted = 7506
tvErrorNoChildID = 7507
tvErrorInvalidIndex = 7508
tvErrorItemNotFound = 7509
tvErrorDuplicateExtension = 7510
tvErrorInvalidExtension = 7511
tvErrorInvalidString = 7512
tvErrorNotImplemented = 7513
tvErrorInvalidEnumValue = 7514
tvErrorCannotMoveToDeleted = 7515
tvErrorPermissionAddNotAllowed = 7516
tvErrorPermissionAddNoUser = 7517
tvErrorPermissionAddNonUser = 7518
tvErrorPermissionAddNewUser = 7519
tvErrorPermissionAddFolderOwner = 7520
tvErrorPermissionAddDuplicateUser = 7521
tvErrorPermissionRemoveNotAllowed = 7522
tvErrorPermissionAccessUseDefault = 7523
tvErrorInvalidLong = 7524
tvErrorDatabaseTimeOut = 7525
tvErrorDatabaseQueryFailed = 7526
tvErrorAudioNoAudioToExport = 7527
tvErrorAudioConversionFailed = 7528
tvErrorAudioInvalidFileFormat = 7529
tvErrorAudioImportNotAllowed = 7530
tvErrorAudioRecordNotAllowed = 7531
tvErrorAudioServerNotRunning = 7532
tvErrorAudioGetTempPath = 7533
tvErrorAudioGetComputerName = 7534
tvErrorAudioGetTempFile = 7535
tvErrorAudioMCICommandFailed = 7536
tvErrorAudioMCICommandAborted = 7537
tvErrorInvalidItemClass = 7538
tvErrorSubItemAlreadyInCollection = 7539
tvErrorAudioInvalidDeviceID = 7540
tvErrorPermissionCannotSetToNone = 7541

tvErrorPermissionDenied = 7542
tvErrorSessionLoggedOff = 7543
tvErrorInvalidDataMember = 7544
tvErrorInvalidTelephonyCharacters = 7545
tvErrorSystemCannotStartServer = 7546
tvErrorSystemCannotShutdownServer = 7547
tvErrorSystemCannotCancelShutdown = 7548
tvErrorCannotSetToNothing = 7549
tvErrorInvalidSNVKCombo = 7550
tvErrorSystemVariableReadOnly = 7551
tvErrorInvalidParentObject = 7552
tvErrorInvalidPassword = 7553
tvErrorInvalidUser = 7554
tvErrorTooManyInvalidLogins = 7555
tvErrorUnsavedObject = 7556
tvErrorDeleteInUseCannotDelete = 7557
tvErrorDeleteInUseCanDelete = 7558
tvErrorPropertyMissing = 7559
tvErrorObjectBeingUpdated = 7560
tvErrorMaxGreetingSpaceExceeded = 7561
tvErrorFailedSendToMailRecipient = 7562
tvErrorDatabaseVersionMismatch = 7563
tvErrorServerVersionMismatch = 7564
tvErrorBuildVersionMismatch = 7565
tvErrorDatabaseConnectionFailed = 7566
tvErrorServerConnectionFailed = 7567
tvErrorServerSyncCallNotAllowed = 7568
tvErrorServerTimeOut = 7569
tvErrorServerUnknown = 7571
tvErrorDatabaseTransactionOpen = 7572
tvErrorDatabaseInvalidCursor = 7573
tvErrorFolderNotFound = 7574
tvErrorParameterRequired = 7575
tvErrorParentSaveBeforeAdd = 7576
tvErrorDatabaseDeadlock = 7577
tvErrorFileCopyFileNotFound = 7578
tvErrorFileCopyDiskFull = 7579
tvErrorAudioImportFailed = 7580
tvErrorServerStationBusy = 7581
tvErrorFileCopyFileAccess = 7582
tvErrorInvalidDevice = 7583
tvErrorServerDBError = 7584
tvErrorServerNoVoxResource = 7585

tvErrorServerFileNotFound = 7586
tvErrorServerDiskFull = 7587
tvErrorServerMaxPartiesExceeded = 7588
tvErrorAudioExportRecording = 7589
tvErrorServerConnectionFailedInvalidNetworkSettings = 7590
tvErrorAudioMCIHardwareProblem = 7591
tvErrorInvalidBoolean = 7592
tvErrorDatabaseOutOfMemory = 7593
tvErrorDatabaseSyntaxError = 7594
tvErrorDatabaseUnknown = 7595
tvErrorAudioExportFailed = 7596
tvErrorAudioCopyFileViaDatabaseFailed = 7597
tvErrorAudioMCIFileProblem = 7598
tvErrorAudioMCIUnknown = 7599
tvErrorInsufficientLicenses = 7600
tvErrorInvalidProcedureCall = 7601
tvErrorCallFeatureNotAvailable = 7602
tvErrorAccountCodeIncomplete = 7603
tvErrorAccountCodeInvalid = 7604
tvErrorAccountCodeRequired = 7605
tvErrorInvalidTransferTarget = 7606
tvErrorObjectDiscarded = 7607
tvErrorServerCantTakeQueueCall = 7608
tvErrorServerFileExists = 7609
tvErrorServerNoAvailableTrunks = 7610
tvErrorServerStationOffHook = 7611
tvErrorInvalidFolder = 7612
tvErrorServerNotRunning = 7613
tvErrorInvalidFile = 7614
tvErrorPasswordRequired = 7615
tvErrorServerAnswerFailed = 7616
tvErrorServerInvalidParty = 7617
tvErrorUnregisteredLicensesExpired = 7618
tvErrorPasswordTooShort = 7619
tvErrorPasswordContainsExtension = 7620
tvErrorPasswordContainsPreventedSequence = 7621
tvErrorPasswordMustBeDifferent = 7622
tvErrorAccountLockedOut = 7623
tvErrorAccountLockedOutDueToPasswordFailure = 7624
tvErrorMustChangePassword = 7625
tvErrorPasswordExpired = 7626
tvErrorServerConferenceFailedNoResources = 7627
tvErrorTrialLicensesExpired = 7628

tvErrorServerNotInstalled = 7629
tvErrorServerDisconnectedParty = 7630
tvErrorServerCantTransfer = 7631
tvErrorServerTargetCantAcceptTransfer = 7632
tvErrorServerTargetCantAcceptTransferBlind = 7633
tvErrorServerTargetCantAcceptTransferSupervised = 7634
tvErrorServerTargetCantAcceptTransferredConferences = 7635
tvErrorServerCantCompleteTransferNoTransferCall = 7636
tvErrorServerCantCompleteTransferNoConsultationCall = 7637
tvErrorServerTransferConvertedSupervisedToBlind = 7638

TVExportFormatType

tvExportFormatTypeXML = 0
TVExportFormatTypeCSV = 1

TVExportType

tvExportTypeView = 0
tvExportTypeData = 1

TVExtensionDialByNameAction

tvExtensionDialByNameActionDoNotList = 0
tvExtensionDialByNameActionListAndPlayExtension = 1
tvExtensionDialByNameActionList = 2

TVFieldType

tvFieldTypeNumber = 0
tvFieldTypeBoolean = 1
tvFieldTypeString = 2
tvFieldTypeDateTime = 3
tvFieldTypeIcon = 4

TVFilterOperator

tvFilterOperatorEqualTo = 0
tvFilterOperatorLessThan = 1
tvFilterOperatorGreaterThan = 2
tvFilterOperatorLike = 3
tvFilterOperatorLessThanOrEqualTo = 4
tvFilterOperatorGreaterThanOrEqualTo = 5
tvFilterOperatorNotEqualTo = 6

TVFinalActionError

tvFinalActionErrorAddressInvalid = 9451
tvFinalActionErrorNoAddress = 9453
tvFinalActionErrorSetAddressToNothing = 9455

TVFolderError

tvFolderErrorInvalidView = 9501
tvFolderErrorUnsavedView = 9502
tvFolderErrorNoView = 9503
tvFolderErrorNoName = 9504
tvFolderErrorDeleteNotAllowed = 9505
tvFolderErrorMoveNotAllowed = 9506
tvFolderErrorRenameNotAllowed = 9507

TVFolderFieldValidation

tvFolderFieldValidationMaxName = 255
tvFolderFieldValidationMinName = 1

TVFoldersError

tvFoldersErrorAddInvalidParent = 9551

TVGreetingError

tvGreetingErrorNonUniqueName = 9001
tvGreetingErrorNoName = 9002
tvGreetingErrorNoAudio = 9003
tvGreetingErrorDeleteDefaultGreeting = 9004
tvGreetingErrorSetDefaultToFalse = 9005
tvGreetingErrorSetActiveToFalse = 9006
tvGreetingErrorDeleteActiveGreeting = 9007

TVGreetingFieldValidation

tvGreetingFieldValidationMaxName = 255
tvGreetingFieldValidationMinName = 1
tvGreetingFieldValidationMaxText = 5000

TVImportContactField

tvImportContactFieldFirstName = 1
tvImportContactFieldLastName = 2
tvImportContactFieldTitle = 3
tvImportContactFieldComment = 4
tvImportContactFieldCompany = 5
tvImportContactFieldBusinessPhone = 6
tvImportContactFieldBusinessPhoneAccessCode = 7
tvImportContactFieldBusinessPhone2 = 8
tvImportContactFieldBusinessPhone2AccessCode = 9
tvImportContactFieldHomePhone = 10
tvImportContactFieldHomePhoneAccessCode = 11
tvImportContactFieldHomePhone2 = 12
tvImportContactFieldHomePhone2AccessCode = 13
tvImportContactFieldMobilePhone = 14
tvImportContactFieldMobilePhoneAccessCode = 15
tvImportContactFieldOtherPhone = 16
tvImportContactFieldOtherPhoneAccessCode = 17
tvImportContactFieldPIN = 18
tvImportContactFieldCallerIDs = 19
tvImportContactFieldAccountCode = 20

TVImportHeaderPosition

tvImportHeaderPositionNone = 0
tvImportHeaderPositionFirstRecord = 1

TVImportOptions

tvImportOptionsReplaceDuplicate = 0
tvImportOptionsAllowDuplicate = 1
tvImportOptionsNoDuplicate = 2

TVInterface

tvInterfaceItem = 0
tvInterfaceAssociate = 1

TVItemChangedBy

tvItemChangedByThisInstance = 0
tvItemChangedByAnotherInstance = 1

TVItemStatus

tvItemStatusNew = 0
tvItemStatusReady = 1
tvItemStatusModified = 2
tvItemStatusDeleted = 3

TVLicenseStatus

tvLicenseStatusUnlicensed = 0
tvLicenseStatusUnregistered = 1
tvLicenseStatusRegistered = 2
tvLicenseStatusRegisteredWithHardwareChange = 3
tvLicenseStatusTrial = 4

TVLicenseType

tvLicenseServer = 0
tvLicenseStation = 1
tvLicenseTrunk = 2
tvLicenseClient = 3
tvLicenseIP = 4
tvLicenseACD = 5
tvLicenseReporter = 6

TVLogCallsLevel

tvLogCallsLevelNone = 0
tvLogCallsLevelInboundOnly = 1
tvLogCallsLevelOutboundOnly = 2
tvLogCallsLevelAll = 3

TVMessageAssociatedPersonType

tvMessageAssociatedPersonTypeSystem = 0
tvMessageAssociatedPersonTypeUser = 1
tvMessageAssociatedPersonTypeContact = 2
tvMessageAssociatedPersonTypeUnknown = 3
tvMessageAssociatedPersonTypeQueue = 4

TVMessageError

tvMessageErrorUnsavedContact = 1501
tvMessageErrorForwardUnsent = 1502
tvMessageErrorReplyUnsent = 1503
tvMessageErrorSaveUnsent = 1504
tvMessageErrorSendExisting = 1505
tvMessageErrorNoRecipient = 1506
tvMessageErrorNoAudio = 1507
tvMessageErrorNoReplyRecipient = 1508
tvMessageErrorAssociateNotAllowed = 1509
tvMessageErrorConfidentialNotAllowed = 1510
tvMessageErrorConfidentialMove = 1511

TVMessageFieldValidation

tvMessageFieldValidationMaxComments = 5000

TVMessageStatus

tvMessageStatusNormal = 0
tvMessageStatusForward = 1
tvMessageStatusReply = 2

TVMessageType

tvMessageTypeVoiceMail = 0
tvMessageTypeCallRecording = 1
tvMessageTypeNthCallRecording = 2
tvMessageTypeSystemCallRecording = 3

TVNameFormat

tvNameFormatFirstLast = 0
tvNameFormatLastFirst = 1
tvNameFormatUserSpecific = 2

TVNotificationError

tvNotificationErrorNoAddress = 2851
tvNotificationErrorAddressTypeMismatch = 2852

TVNotificationFieldValidation

tvNotificationFieldValidationMaxLevel = 2
tvNotificationFieldValidationMaxMessageAction = 3

TVNotificationLevel

tvNotificationLevelNone = 0
tvNotificationLevelAll = 1
tvNotificationLevelUrgentOnly = 2

TVNotificationMessageAction

tvNotificationMessageActionDoNotAttach = 0
tvNotificationMessageActionAttach = 1
tvNotificationMessageActionAttachAndMarkHeard = 2
tvNotificationMessageActionAttachAndDelete = 3

TVNotificationType

tvNotificationTypeEmail = 1
tvNotificationTypePager = 2
tvNotificationTypeCall = 3

TVObjectClass

tvClassSession = 1
tvClassFolder = 10
tvClassSystem = 11
tvClassPermission = 20
tvClassAddress = 21
tvClassSetting = 22
tvClassView = 23
tvClassShortcut = 24
tvClassSchedule = 27
tvClassScheduledDay = 28
tvClassScheduledDate = 29
tvClassVariable = 33
tvClassNotification = 34
tvClassParty = 35
tvClassAudioClip = 36
tvClassField = 41
tvClassColumn = 42
tvClassNotifyScheduleItem = 43
tvClassRoutingListAction = 44
tvClassRoutingListFinalAction = 45
tvClassReportingCategory = 46
tvClassSystemSetting = 47
tvClassShortcutGroup = 48
tvClassStationType = 49
tvClassStationFeature = 50
tvClassStationParameter = 51
tvClassAgent = 101
tvClassAgentStat = 2301
tvClassAutoAttendant = 201
tvClassCall = 301
tvClassCallHistory = 401
tvClassCallHistoryParty = 402
tvClassCallRule = 1001
tvClassContact = 501
tvClassDeviceStation = 601
tvClassGreeting = 1501
tvClassIVRPlugIn = 1401
tvClassMessage = 701
tvClassPersonalStatus = 1601
tvClassQueue = 901
tvClassQueueStat = 2201
tvClassRoutingList = 1801

tvClassServiceInternet = 1101
tvClassServicePhone = 1102
tvClassServicePhoneGateway = 1103
tvClassServiceSwitch = 1104
tvClassServiceSwitchGateway = 1105
tvClassServiceRouting = 1106
tvClassSystemTarget = 1701
tvClassUser = 1301
tvClassUserGroup = 1302
tvClassRole = 1302
tvClassWorkgroup = 2101

TVObjectDialingServiceClass

tvDialingServiceClassPhone = 1102
tvDialingServiceClassSwitch = 1104
tvDialingServiceClassInternet = 1101
tvDialingServiceClassPhoneGateway = 1103
tvDialingServiceClassSwitchGateway = 1105
tvDialingServiceClassRouting = 1106

TVObjectItemClass

tvItemClassNoItem = 0xffffffff
tvItemClassDeletedItem = 0
tvItemClassAgentItem = 100
tvItemClassAutoAttendantItem = 200
tvItemClassCallItem = 300
tvItemClassCallHistoryItem = 400
tvItemClassContactItem = 500
tvItemClassDeviceItem = 600
tvItemClassMessageItem = 700
tvItemClassQueueItem = 900
tvItemClassCallRuleItem = 1000
tvItemClassServiceItem = 1100
tvItemClassUserItem = 1300
tvItemClassIVRPlugInItem = 1400
tvItemClassGreetingItem = 1500
tvItemClassPersonalStatusItem = 1600
tvItemClassSystemTargetItem = 1700
tvItemClassRoutingListItem = 1800
tvItemClassWorkgroupItem = 2100
tvItemClassQueueStatItem = 2200
tvItemClassAgentStatItem = 2300

TVOutboundCallerIDPresentation

tvOutboundCallerIDPresentationSystemDefault = 0
tvOutboundCallerIDPresentationNetwork = 1
tvOutboundCallerIDPresentationAvailable = 2
tvOutboundCallerIDPresentationRestricted = 3

TVPartyDirection

tvPartyDirectionInbound = 0
tvPartyDirectionOutbound = 1

TVPartyFeature

tvPartyFeatureAnnounce = 0
tvPartyFeatureAnswer = 1
tvPartyFeatureAssociate = 2
tvPartyFeatureDisconnect = 7
tvPartyFeatureHold = 8
tvPartyFeatureMute = 10
tvPartyFeatureRoleCoach = 16
tvPartyFeatureRoleMonitor = 17
tvPartyFeatureRolePeer = 18
tvPartyFeatureRolePupil = 19
tvPartyFeatureSetRole = 22

TVPartyHistoryResult

tvPartyHistoryResultNone = 0
tvPartyHistoryResultHungUp = 1
tvPartyHistoryResultHungUpUpon = 2
tvPartyHistoryResultAbandoned = 3
tvPartyHistoryResultVoiceMail = 4
tvPartyHistoryResultBlindTransfer = 5
tvPartyHistoryResultSupervisedTransfer = 6
tvPartyHistoryResultMerged = 7
tvPartyHistoryResultLoggedIn = 8
tvPartyHistoryResultDirectVoiceMail = 9
tvPartyHistoryResultNoAnswer = 10
tvPartyHistoryResultAbandonedUpon = 11
tvPartyHistoryResultLoginFailed = 12

TVPartyRole

tvPartyRolePeer = 0
tvPartyRoleMonitor = 1
tvPartyRolePupil = 2
tvPartyRoleCoach = 3

TVPartyStatus

`_tvPartyStatusMax = 38`
`tvPartyStatusUnknown = 37`
`tvPartyStatusNone = 0`
`tvPartyStatusActive = 1`
`tvPartyStatusAlerting = 29`
`tvPartyStatusApplication = 33`
`tvPartyStatusAttempting = 28`
`tvPartyStatusAttemptingTransfer = 35`
`tvPartyStatusCalling = 23`
`tvPartyStatusDialing = 30`
`tvPartyStatusDisconnected = 3`
`tvPartyStatusGreetingListening = 26`
`tvPartyStatusGreetingPlaying = 6`
`tvPartyStatusHold = 4`
`tvPartyStatusMessageLeaving = 8`
`tvPartyStatusMessageScreening = 9`
`tvPartyStatusMessageTaking = 7`
`tvPartyStatusOffering = 5`
`tvPartyStatusParked = 25`
`tvPartyStatusQueued = 27`
`tvPartyStatusRinging = 2`
`tvPartyStatusTransferring = 34`
`tvPartyStatusWaiting = 11`

TVPermissionAccessLevel

`tvPermissionAccessLevelUnknown = 0xffffffff`
`tvPermissionAccessLevelNone = 0`
`tvPermissionAccessLevelReadOnly = 1`
`tvPermissionAccessLevelFull = 2`

TVPermissionType

`tvPermissionTypeBoolean = 0`
`tvPermissionTypeAccessLevel = 1`

TVPersonalStatusCallForwarding

`tvPersonalStatusCallForwardingUseDefault = 0`
`tvPersonalStatusCallForwardingSetOn = 1`
`tvPersonalStatusCallForwardingSetOff = 2`

TVPersonalStatusError

tvPersonalStatusErrorNonUniqueName = 8501
tvPersonalStatusErrorNoName = 8502
tvPersonalStatusErrorNoCategory = 8503
tvPersonalStatusErrorInvalidCallForwarding = 8504
tvPersonalStatusErrorCannotDeleteLAPS = 8505
tvPersonalStatusErrorSystemDefinedStatus = 8506

TVPersonalStatusFieldValidation

tvPersonalStatusFieldValidationLengthMaxDescription = 100
tvPersonalStatusFieldValidationLengthMaxName = 50

TVPersonalStatusType

tvPersonalStatusTypeAvailable = 0
tvPersonalStatusTypeDoNotDisturb = 1
tvPersonalStatusTypeInAMeeting = 2
tvPersonalStatusTypeOutOfTheOffice = 3
tvPersonalStatusTypeVacation = 4
tvPersonalStatusTypeAvailableACDOnly = 5
tvPersonalStatusTypeAvailableNonACD = 6
tvPersonalStatusTypeBreak = 7
tvPersonalStatusTypeCustom = 8

TVRecipientError

tvRecipientErrorAddNotAllowed = 1950
tvRecipientErrorDuplicateRecipient = 1951
tvRecipientErrorDiscardNotAllowed = 1952
tvRecipientErrorRemoveNotAllowed = 1953
tvRecipientErrorSaveNotAllowed = 1954
tvRecipientErrorAddInvalidObject = 1955
tvRecipientErrorAddNoMailbox = 1956

TVRegistryDataType

tvRegistryDataTypeBoolean = 0
tvRegistryDataTypeLong = 1
tvRegistryDataTypeString = 2

TVRegistryLocation

TVRegistryLocationDatabasePerUser = 0
tvRegistryLocationDatabasePerSystem = 1
tvRegistryLocationRegistryCurrentUser = 2
tvRegistryLocationRegistryLocalMachine = 3

TVRoutingListActionError

tvRoutingListActionErrorNoAddress = 9401
tvRoutingListActionErrorSetAddressToNothing = 9402
tvRoutingListActionErrorSetPromptCallerGreeting = 9403
tvRoutingListActionErrorNoVoiceTitle = 9404
tvRoutingListActionErrorInvalidRingDuration = 9405
tvRoutingListActionErrorCannotForward = 9406

TVRoutingListActionType

tvRoutingListActionTypeCallMeWhereIAm = 1
tvRoutingListActionTypeCallAddress = 2
tvRoutingListActionTypePlayGreeting = 3
tvRoutingListActionTypePause = 4

TVRoutingListError

tvRoutingListErrorDeleteDefault = 9251
tvRoutingListErrorNoName = 9252
tvRoutingListErrorNonUniqueName = 9253
tvRoutingListErrorSetDefaultToFalse = 9254
tvRoutingListErrorEditSystemRL = 9255
tvRoutingListErrorSetActiveToFalse = 9256
tvRoutingListErrorDeleteActive = 9257

TVRoutingListFieldValidation

tvRoutingListFieldValidationMaxName = 128
tvRoutingListFieldValidationMinName = 1

TVRoutingListFinalActionType

tvRoutingListFinalActionTypeTakeMessage = 1
tvRoutingListFinalActionTypeDisconnect = 2
tvRoutingListFinalActionTypeTransferToExtension = 3
tvRoutingListFinalActionTypePauseAndRestart = 4
tvRoutingListFinalActionTypePlayBusySignal = 5
tvRoutingListFinalActionTypeTransferToVMail = 6
tvRoutingListFinalActionTypeHangup = 7

TVRoutingListPlayGreetingType

tvRoutingListPlayGreetingTypeDoNotPlay = 0
tvRoutingListPlayGreetingTypeSystemHoldGreeting = 1
tvRoutingListPlayGreetingTypeCustomGreeting = 2

TVRoutingListPromptCallerType

tvRoutingListPromptCallerTypeDoNotPrompt = 0
tvRoutingListPromptCallerTypeSystemGreeting = 1
tvRoutingListPromptCallerTypeCustomGreeting = 2

TVScheduledDaysWeekday

tvScheduledDaysWeekdaySunday = 1
tvScheduledDaysWeekdayMonday = 2
tvScheduledDaysWeekdayTuesday = 3
tvScheduledDaysWeekdayWednesday = 4
tvScheduledDaysWeekdayThursday = 5
tvScheduledDaysWeekdayFriday = 6
tvScheduledDaysWeekdaySaturday = 7

TVScheduleItemFieldValidation

tvScheduleItemFieldValidationMaxType = 5

TVScheduleItemType

tvScheduleItemTypeBusinessHours = 0
tvScheduleItemTypeNonBusinessHours = 1
tvScheduleItemTypeAfterHoursWorkDays = 2
tvScheduleItemTypeNonWorkDays = 3
tvScheduleItemTypeCustomHours = 4
tvScheduleItemTypeHolidays = 5

TVSchedulesError

tvSchedulesErrorDuplicateName = 2801

TVSearchType

tvSearchDefault = 0
tvSearchID = 1
tvSearchOrdinal = 2
tvSearchKey = 3

TVServerConnectionLevel

tvServerConnectionLevelBasic = 0xffffffff
tvServerConnectionLevelFull = 0

TVServerStatus

tvServerLicenseError = 6
tvServerStarting = 2
tvServerStarted = 3
tvServerShutDownScheduled = 5
tvServerStopping = 4
tvServerStopped = 1
tvServerUnknown = 0

TVSessionStatus

tvSessionStatusLoggedOff = 0
tvSessionStatusLoggedOn = 1
tvSessionStatusLoggedOnWithoutCalls = 2

TVSessionValidateLogonResult

tvSessionValidateLogonResultSuccess = 0
tvSessionValidateLogonResultInvalidUser = 1
tvSessionValidateLogonResultInvalidPassword = 2
tvSessionValidateLogonResultInvalidStation = 3

TVShortcutError

tvShortcutErrorGroupNameRequired = 4401
tvShortcutErrorShortcutNameandFolderRequired = 4402

TVShortcutFieldValidation

tvShortcutFieldValidationMaxName = 255
tvShortcutFieldValidationMinName = 1

TVShortcutGroupFieldValidation

tvShortcutGroupFieldValidationMaxName = 255
tvShortcutGroupFieldValidationMinName = 1

TVShortcutGroupIconType

tvShortcutGroupIconTypeLarge = 0
tvShortcutGroupIconTypeSmall = 1

TVSortOrder

tvSortToggle = 0
tvSortAscending = 1
tvSortDescending = 2

TVSpecialUsers

tvEveryOne = 0xffffffff

TVStationAnalogPhoneType

tvStationAnalogPhoneTypeStandard = 0
tvStationAnalogPhoneTypeAastra = 1
tvStationAnalogPhoneTypeCybiolink = 2

TVStationButtonFieldValidation

tvStationButtonFieldValidationMinRingDelay = 0xffffffff
tvStationButtonFieldValidationMaxRingDelay = 999

TVStationError

tvStationErrorFeatureRequired = 2501
tvStationErrorStationRequired = 2502
tvStationErrorButtonAlreadyDefined = 2503
tvStationErrorAddressRequired = 2504
tvStationErrorButtonSetAddressToNothing = 2505

TVStationFeatureParameterType

tvStationFeatureParameterTypeNone = 0xffffffff
tvStationFeatureParameterTypeAddress = 0
tvStationFeatureParameterTypeStationNumber = 1
tvStationFeatureParameterTypeRingDelay = 2

TVStationFieldValidation

tvStationFieldValidationMaxDefaultIdleString = 16
tvStationFieldValidationMaxDirectConnectDialString = 255
tvStationFieldValidationMinCadenceID = 1
tvStationFieldValidationMaxCadenceID = 16
tvStationFieldValidationMinDefaultVolume = 1
tvStationFieldValidationMaxDefaultVolume = 21

TVStationPadCharacter

tvStationPadCharacterPound = 35
tvStationPadCharacterStar = 42
tvStationPadCharacterZero = 48
tvStationPadCharacterOne = 49
tvStationPadCharacterTwo = 50
tvStationPadCharacterThree = 51
tvStationPadCharacterFour = 52
tvStationPadCharacterFive = 53
tvStationPadCharacterSix = 54
tvStationPadCharacterSeven = 55
tvStationPadCharacterEight = 56
tvStationPadCharacterNine = 57

TVStationPadExtension

tvStationPadExtensionBefore = 0
tvStationPadExtensionAfter = 1
tvStationPadExtensionDoNotPad = 2

TVStationPhoneFeatures

tvStationPhoneFeaturesNone = 0
tvStationPhoneFeaturesCID = 1
tvStationPhoneFeaturesCallWaitingCID = 2
tvStationPhoneFeaturesMessageIndicator = 4
tvStationPhoneFeaturesStutter = 8

TVStationTransferMode

tvStationTransferModeDirect = 0
tvStationTransferModeAssisted = 1

TVStationUsage

tvStationUsageOwner = 0
tvStationUsageOwnerAndForwardCalls = 1
tvStationUsageVisitor = 2

TVStoreHistoryType

tvStoreHistoryNone = 0
tvStoreHistoryExternalOnly = 1
tvStoreHistoryAll = 2

TVSystemTargetAvailability

tvSystemTargetAvailabilityAvailable = 0
tvSystemTargetAvailabilityInCall = 1
tvSystemTargetAvailabilityRinging = 2
tvSystemTargetAvailabilityWrapUp = 3

TVSystemTargetCallForwardingType

tvSystemTargetCallForwardingTypeNone = 0
tvSystemTargetCallForwardingTypeInternal = 1
tvSystemTargetCallForwardingTypeExternal = 2

TVSystemTargetError

tvSystemTargetErrorCannotDelete = 8751
tvSystemTargetErrorCannotDiscard = 8752
tvSystemTargetErrorCannotMove = 8753
tvSystemTargetErrorCannotSave = 8754

TVSystemTargetType

tvSystemTargetTypeAutoAttendant = 201
tvSystemTargetTypeIVRPlugIn = 1401
tvSystemTargetTypeQueue = 901
tvSystemTargetTypeUser = 1301
tvSystemTargetTypeWorkgroup = 2101

TVUserAccountCodeBehavior

tvUserAccountCodeBehaviorNone = 0
tvUserAccountCodeBehaviorRequire = 1
tvUserAccountCodeBehaviorVerify = 2

TVUserAnnounceCallBehavior

tvUserAnnounceCallBehaviorNone = 0
tvUserAnnounceCallBehaviorCallerInternal = 1
tvUserAnnounceCallBehaviorCallerExternal = 2
tvUserAnnounceCallBehaviorCallerExternalDirect = 4
tvUserAnnounceCallBehaviorCallee = 8
tvUserAnnounceCallBehaviorTransferrer = 16

TVUserCallWaiting

tvUserCallWaitingDisabled = 0
tvUserCallWaitingEnabled = 1
tvUserCallWaitingEnabledWithoutBeep = 2

TVUserCategory

tvUserCategoryLocalUser = 0
tvUserCategoryACDUser = 1
tvUserCategoryRemoteUser = 3

TVUserDefaultContactAction

tvUserDefaultContactActionOpen = 0
tvUserDefaultContactActionCall = 1

TVUserEmptyDeletedBehavior

tvUserEmptyDeletedBehaviorNone = 0
tvUserEmptyDeletedBehaviorOnExit = 1
tvUserEmptyDeletedBehaviorInterval = 2

TVUserError

tvUserErrorInvalidDIDNumber = 2001
tvUserErrorNoLastName = 2002
tvUserErrorServerSettingsMissing = 2003
tvUserErrorDuplicateFullName = 2004
tvUserErrorNotInAddresses = 2005
tvUserErrorInvalidDefaultAddress = 2006
tvUserErrorPasswordFailNoMatch = 2009
tvUserErrorCannotDeleteSystemUser = 2012
tvUserErrorCannotRenameSystemUser = 2013
tvUserErrorPasswordInvalidCharacters = 2014
tvUserErrorNonSystemSchedule = 2015
tvUserErrorInvalidCallForwarding = 2016

TVUserExternalCallTypes

tvUserExternalCallTypesNone = 0
tvUserExternalCallTypesLocal = 1
tvUserExternalCallTypesLocalLong = 2
tvUserExternalCallTypesLocalLongIntl = 3

TVUserFieldValidation

tvUserFieldValidationMaxComment = 5000
tvUserFieldValidationMaxCompany = 50
tvUserFieldValidationMaxDIDList = 500
tvUserFieldValidationMaxExchangeMailbox = 5000
tvUserFieldValidationMinExtension = 1
tvUserFieldValidationMaxExtension = 5
tvUserFieldValidationMaxFirstName = 64
tvUserFieldValidationMaxJobTitle = 30
tvUserFieldValidationMaxLastName = 30
tvUserFieldValidationMaxPassword = 12
tvUserFieldValidationMaxExtensionEx = 10
tvUserFieldValidationMinConferenceMembers = 0xffffffff
tvUserFieldValidationMinGreetingTitleSize = 1
tvUserFieldValidationMinMaxMessageDuration = 6
tvUserFieldValidationMaxMaxMessageDuration = 300
tvUserFieldValidationMinMaxMailboxSize = 0
tvUserFieldValidationMaxMaxMailboxSize = 59940
tvUserFieldValidationMaxMaxMailboxSizeEx = 0x00092784

TVUserInboundCallBehavior

tvUserInboundCallBehaviorNone = 0
tvUserInboundCallBehaviorDisplayCallMonitor = 1
tvUserInboundCallBehaviorPlayCallerName = 2
tvUserInboundCallBehaviorFlashTitleBar = 4
tvUserInboundCallBehaviorPlayCalleeName = 8
tvUserInboundCallBehaviorFlashCallMonitorTab = 16

TVUserMessageOrder

tvUserMessageOrderUnheardLIFOHeardLIFO = 1
tvUserMessageOrderUnheardLIFOHeardFIFO = 2
tvUserMessageOrderUnheardFIFOHeardFIFO = 4
tvUserMessageOrderUnheardFIFOHeardLIFO = 8
tvUserMessageOrderAllFIFO = 16
tvUserMessageOrderAllLIFO = 32

TVUserNameFormat

tvUserNameFormatFirstLast = 0
tvUserNameFormatLastFirst = 1

TVUserOutboundCallBehavior

tvUserOutboundCallBehaviorNone = 0
tvUserOutboundCallBehaviorDisplayMonitorOnNew = 1
tvUserOutboundCallBehaviorDisplayMonitorOnReturn = 2

TVUserPromptCallerForName

tvUserPromptCallerForNameNever = 0
tvUserPromptCallerForNameWhenNoVoiceTitle = 1
tvUserPromptCallerForNameWhenNoCallerID = 2

TVUserRingPattern

tvUserRingPatternTwoSecond = 1
tvUserRingPatternOneSecond = 2
tvUserRingPatternSplash1 = 3
tvUserRingPatternSplash2 = 4
tvUserRingPatternSplash3 = 5
tvUserRingPatternLongShort = 6
tvUserRingPatternShortLong = 7

TVUserSendDigitsBehavior

tvUserSendDigitsBehaviorNone = 0
tvUserSendDigitsBehaviorDID = 1
tvUserSendDigitsBehaviorExtension = 2

TVUserStrataRingPattern

tvUserStrataRingPatternLowTone = 1
tvUserStrataRingPatternMediumTone = 2
tvUserStrataRingPatternHighTone = 3
tvUserStrataRingPatternCombinedTone = 4
tvUserStrataRingPatternSingleTone = 5

TVViewError

tvViewErrorResetNotAllowed = 9651
tvViewErrorSetCurrentToFalse = 9652
tvViewErrorSortStringInvalid = 9653
tvViewErrorFilterStringInvalid = 9654
tvViewErrorFilterValueInvalid = 9655
tvViewErrorFieldNameInvalid = 9656

TVViewFieldValidation

tvViewFieldValidationMaxFilter = 255
tvViewFieldValidationMaxSort = 255

TVWorkgroupErrors

tvWorkgroupErrorMissingName = 1701
tvWorkgroupErrorInvalidRingDuration = 1702
tvWorkgroupErrorNonUniqueName = 1703

TVWorkgroupFieldValidation

tvWorkgroupFieldValidationMaxComments = 5000
tvWorkgroupFieldValidationMinName = 0
tvWorkgroupFieldValidationMaxName = 30
tvWorkgroupFieldValidationMaxDID = 500
tvWorkgroupFieldValidationMaxExtension = 4

TVWorkgroupMembersError

tvWorkgroupMembersErrorAlreadyExists = 1801
tvWorkgroupMembersErrorInvalidPerson = 1802
tvWorkgroupMembersErrorInvalidOwner = 1803

TVWorkgroupRoutingType

tvWorkgroupRoutingTypeSimultaneous = 0
tvWorkgroupRoutingTypeRoundRobin = 2
tvWorkgroupRoutingTypeSequential = 4

INDEX

Symbols

* telephone commands (table), 6-6

A

about

- in-band signaling, 6-2
- IVR Plug-in API, 3-2
- TeleVantage Client API, 2-2
- TeleVantage Device Status API, 4-2
- TeleVantage SDK, 1-2
- TeleVantage TAPI Service Provider, 5-2

Address object, D-3

Addresses object, D-3

Agent object, D-4

Agent object group, 2-11

Agents object, D-5

AgentStat object, D-5

Application Programming Interface (API)

defined, 1-3

AudioClip object, D-6

C

Call object, D-7

Call object group, 2-12

CallHistory object, D-9

CallHistory object group, 2-12

CallRule object, D-10

CallRule object group, 2-13

class

defined, 1-3

Client API. *See* TeleVantage Client API

Column object, D-11

Columns object, D-11

component

defined, 1-3

Component Object Model (COM)

defined, 1-3

Contact object, D-12

Contact object group, 2-13

D

definitions of terms used in this manual, 1-3

development tools, 1-3

Device Status API. *See* TeleVantage Device Status API

Dialing service object group, 2-14

E

Error object, D-13

F

Field object, D-13

Fields object, D-13

Folder object, D-14

Folder object group, 2-7

folders

default folder structure, 2-9

types, 2-10

used in TeleVantage Client API, 2-7

Folders object, D-15

G

Greeting object, D-15

Greeting object group, 2-15

I

IAssociate object, D-16

ICOMSecurity object, D-16

IDialingService object, D-17

IItem object, D-17

installing

TeleVantage Client API, 1-3

TeleVantage Device Status API, 1-3

TeleVantage IVR Plug-in API, 1-3

InternetService object, D-18

IPhoneService object, D-18

Items object, D-19

IVR Plug-in API. *See* TeleVantage IVR Plug-in API

IVR Plug-ins

debugging in Visual Basic, 3-8

licensing, 3-3

L

LocaleCode object, D-19

LocaleCodes object, D-20

M

Message object, D-20

Message object group, 2-15

Microsoft Visual Basic

debugging IVR Plug-ins with, 3-8

starting new TeleVantage Client API project, 2-20
module
defined, 1-3

N

Notification object, D-21
Notification object group, 2-16
NotifyScheduleItem object, D-21
NotifyScheduleItems object, D-22

O

Object

Address, D-3
Addresses, D-3
Agent, D-4
Agents, D-5
AgentStat, D-5
AudioClip, D-6
Call, D-7
CallHistory, D-9
CallRule, D-10
Column, D-11
Columns, D-11
Contact, D-12
Error, D-13
Field, D-13
Fields, D-13
Folder, D-14
Folders, D-15
Greeting, D-15
IAssociate, D-16
ICOMSecurity, D-16
IDialingService, D-17
IItem, D-17
InternetService, D-18
IPhoneService, D-18
Items, D-19
LocaleCode, D-19
LocaleCodes, D-20
Message, D-20
Notification, D-21
NotifyScheduleItem, D-21
NotifyScheduleItems, D-22
Parties, D-22
Party, D-23
PartyHistories, D-24
PartyHistory, D-24

Permission, D-25
Permissions, D-25
PersonalStatus, D-26
PhoneGatewayService, D-27
PhoneService, D-28
QueueByPeriodStat, D-29
QueueByShiftStat, D-30
QueueStat, D-30
Recipients, D-32
Role, D-32
Roles, D-32
RoutingList, D-33
RoutingListAction, D-34
RoutingListActions, D-34
RoutingListFinalAction, D-35
RoutingService, D-35
Schedule, D-36
ScheduledDate, D-36
ScheduledDates, D-36
ScheduledDay, D-37
ScheduledDays, D-37
Schedules, D-38
Session, D-38
Shortcut, D-43
ShortcutGroup, D-44
ShortcutGroups, D-44
Shortcuts, D-45
Station, D-45
StationButton, D-46
StationButtons, D-46
StationFeature, D-47
StationFeatures, D-47
StationParameter, D-48
StationParameters, D-48
StationType, D-48
StationTypes, D-49
SwitchGatewayService, D-49
SwitchService, D-50
System, D-50
SystemSetting, D-51
SystemSettings, D-52
SystemTarget, D-52
User, D-53
View, D-55
Views, D-56
Workgroup, D-56
WorkgroupMember, D-57
WorkgroupMembers, D-57

object

- defined, 1-3
- object group diagrams
 - key, 2-3
- object groups
 - Agent, 2-11
 - Call, 2-12
 - CallHistory, 2-12
 - CallRule, 2-13
 - Contact, 2-13
 - Dialing services, 2-14
 - Folder, 2-7
 - Greeting, 2-15
 - Message, 2-15
 - Notification, 2-16
 - RoutingList, 2-17
 - Schedule, 2-18
 - Session, 2-4
 - Station, 2-19
 - StationFeature, 2-19
 - System, 2-5
 - User, 2-6
 - Workgroup, 2-20
- object groupsLPersonalStatus, 2-16
- object structures
 - in TeleVantage Client API, 2-2

P

- Parties object, D-22
- Party object, D-23
- PartyHistories object, D-24
- PartyHistory object, D-24
- Permission object, D-25
- Permissions object, D-25
- PersonalStatus object, D-26
- PersonalStatus object group, 2-16
- PhoneGatewayService object, D-27
- PhoneService object, D-28
- programming tips, 2-21

Q

- QueueByPeriodStat object, D-29
- QueueByShiftStat object, D-30
- QueueStat object, D-30

R

- Recipients object, D-32

- Role object, D-32
- Roles object, D-32
- RoutingList object, D-33
- RoutingList object group, 2-17
- RoutingListAction object, D-34
- RoutingListActions object, D-34
- RoutingListFinalAction object, D-35
- RoutingService object, D-35

S

- Schedule object, D-36
- Schedule object group, 2-18
- ScheduledDate object, D-36
- ScheduledDates object, D-36
- ScheduledDay object, D-37
- ScheduledDays object, D-37
- Schedules object, D-38
- Session object, D-38
- Session object group, 2-4
- Shortcut object, D-43
- ShortcutGroup object, D-44
- ShortcutGroups object, D-44
- Shortcuts object, D-45
- starting
 - new project, 2-20
- Station object, D-45
- Station object group, 2-19
- StationButton object, D-46
- StationButtons object, D-46
- StationFeature object, D-47
- StationFeature object group, 2-19
- StationFeatures object, D-47
- StationParameter object, D-48
- StationParameters object, D-48
- StationType object, D-48
- StationTypes object, D-49
- SwitchGatewayService object, D-49
- SwitchService object, D-50
- System object, D-50
- System object group, 2-5
- SystemSetting object, D-51
- SystemSettings object, D-52
- SystemTarget object, D-52

T

telephone commands

- * commands (table), 6-6
- call handling menu commands (table), 6-3
- quick call menu commands (table), 6-4
- quick call menu commands for call center agents (table), 6-6
- Voice Mail.Account menu commands (table), 6-5

TeleVantage

- using in-band signaling with, 6-2

TeleVantage Client API

- about, 2-2
- installing, 1-3
- key to object group diagrams, 2-3
- object structures, 2-2
- programming tips and examples, 2-21
- starting new project, 2-20

TeleVantage Device Status API

- about, 4-2
- Device Status components, 4-2
- installing, 1-3
- quick reference, 4-5
- sample project, 4-2

TeleVantage IVR Plug-in API

- about, 3-2
- components, B-2
- configuring, 3-4
- installing, 1-3
- monitoring standard and custom call data, 3-3
- quick reference, 3-9
- using
 - sample Customer ID Plug-in, 3-6

Televantage SDK

- about, 1-2

TeleVantage TAPI Service Provider

- about, 5-2
- accessing custom call data, 5-3
- bearer modes supported, 5-3
- call states, 5-3
- capabilities, 5-2
- exported TSP functions, 5-3
- TAPI functions supported, 5-2

terms used in this manual, 1-3

type library

- defined, 1-3

U

- User object, D-53

- User object group, 2-6

- using in-band signaling With TeleVantage
 - about, 6-2

V

- View object, D-55

- Views object, D-56

W

- Workgroup object, D-56

- Workgroup object group, 2-20

- WorkgroupMember object, D-57

- WorkgroupMembers object, D-57