ZOIPER WEB API

REVISION: *Zoiper Web 2.7*
UPDATED: *November 2011*

CONTENTS

Introduction

Zoiper Web is designed to be embedded in web pages and is compatible with the most common web browsers on the market. It allows the website visitor to make and receive VoIP calls using the SIP or IAX2 protocols. It also allows the website to control the phone by using the Zoiper Web API, using JavaScript or similar technologies. Currently Zoiper Web supports a wide variety of browsers on Microsoft Windows operating system including, but not limited to:

Microsoft Internet Explorer 6 and above, Mozilla Firefox 3.0 and above, Safari . NPAPI support is no longer active on Chrome.

Both the Microsoft ActiveX and NPAPI (Netscape Plugin Application Programming Interface) technologies are supported, offering native plugins for every browser technology.

NB: Some of the features in this document are marked as **"not supported in all versions"**. This means that they are not available with the standard Zoiper WEB version which is for sale on the Zoiper online shop. For more information regarding the availability of these features and how you can purchase them, please contact sales@zoiper.com

1. Installation

Zoiper Web is using an installer program to install itself on the user's PC. This program takes care for the Internet Explorers CLSID registration and for setup of the NPAPI registry keys. This way a single installation can be used to enable all the browsers on the user's PC to use Zoiper Web, without having to install it for every browser separately.

For internet explorer, a cabinet (.cab) file which encapsulates the installer is provided. This enables Internet Explorer to automatically download and run the installer for the user. (This will also install Zoiper Web for all other browsers on the user's pc). The class ID (CLSID) used to identify Zoiper Web in Internet Explorer and Microsoft Windows is {BCCA9B64-41B3-4A20-8D8BE69FE61F1F8B}.

For Firefox users, a standalone executable installer is available. There is no .xpi installer.The MIME type used to identify Zoiper Web in NPAPI based browsers is "application/x-zoiper-plugin".

2. Zoiper Web API interface

The API provided with Zoiper Web consists of several classes which provide the functionality of creating and managing configuration, accounts, contacts and calls. It also provides event feedback from Zoiper Web to the browser in the form of predefined callback functions in the web page that is using it.

All the properties, method and callback parameters are using the "string" type. Note that because there are boolean values that can be passed and all the values are actually strings, there are two strings values representing "true" and "false" that are used in that case.

The structure of the API is based on one main class named "Phone" which provides you with the facilities to obtain and manage "Config", "Account", "Call" and "Contact" objects, and a predefined number of named functions, called "Callbacks" or "Events", that must be defined in the JavaScript script so that Zoiper can call them back. There is a certain "Callback" which notifies the user that the phone has finished starting up and provides a reference to a "Phone" class instance. This "Callback" is used as an entry point for the Zoiper Web usage. The "Callbacks" are not really part of the classes but they will be described along with them as they are logically related.

3. The "Phone" class

The "Phone" class represents the Zoiper Web as a whole. It provides facilities for general configuration (“Config” object), creation and management of "Account", "Call" and "Contact" objects.

This is the main class of Zoiper Web and it is the only object that the user can directly refer to from the JavaScript itself (from the <object> or <embed> tags). Using the object obtained that way is not advisable because it cannot be used before the "OnZoiperReady". A certain "Callback" is called to indicate that the "Phone" is ready to use. This "Callback" also returns as parameter the reference to the "Phone" object to be used. Obtaining the "Phone" reference has another advantage, it returns the right object reference to use, no matter if the Zoiper Web instance is loaded from <object> or <embed> tag, thus making browser or technology independent.

The properties and methods of the "Phone" class can be organized in a several groups as described below:

Call management properties and methods

- *property* **Call:**
  - Read only.
  - Gets the class reference to the current active call "Call" object. If there is no active call, "null" is returned.
- *property* **NumberOfCalls:**
  - Read only.
  - Gets the number of current calls.
- *method* **Dial(***number***):**
  - Return "Call" object reference.
  - Create a new "Call" object and dials a desired phone number/extension.
  - Dial the number provided as the parameter "*number*". Any current active call is put on hold and the new call becomes the active one.
  - The call is made for the current active "Account". If there is no active "Account", the first "Account" in the list will be used.
  - If the call creation is unsuccessful or there are no accounts at all, the method will return "null".
- *method* **Hang():**

- Return no value.
- Hang up the active call.

- *method* **ShowTransferDialog():**
  - Return value can be “True” or “False”.
  - Shows the transfer dialog if there is an active call that could be transferred.

Account management

- *method* **UseAccount(***name***):**
  - Return "Account" object reference.
  - Select which account to be set as default provided as the parameter "name".
  - If there is no account with the provided "name", the method will return "null".
- *method* **GetAccount(***name***):**
  - Return "Account" object reference.
  - Get the “Account” object reference by its “name” provided as parameter.
  - If there is no account with the provided "name", the method will return "null".
- *method* **AddAccount(***name, protocol***):**
  - Return "Account" object reference.
  - Creates new account or get an existing one with the provided as parameter "name".
  - The new account will be using the provided as parameter "protocol".
  - The "protocol" parameter can take the following values "SIP", "IAX" or “XMPP”.
  - If the account cannot be created, the method will return "null".
  - The "name" of the account is used as unique identifier.
  - * “XMPP” “Account” creation is not supported in all versions.
- *method* **DelAccount(***name***):**
  - Return no value.
  - Delete the account with the provided as parameter "name" (if it exists).
- *method* **Login(***username, password***):**
  - Return no value.
  - Login to Zoiper Web to the Zoiper Service by using the provided as parameters "username" and "password".
- *method* **Logout():**
  - Return no value

- Logout from Zoiper Web to the Zoiper Service.

Contacts management

- *method* **GetContact(***phone***)**:
  - Return "Contact" object reference.
  - Find an existing contact by its primary phone number provided as parameter "phone".
  - If contact with the provided “phone” does not exist, the method will return "null".
- *method* **GetContactByIndex(***index***)**:
  - Return "Contact" object reference.
  - Find an existing contact by its “index”. This method is used to enumerate all the contacts. The “index” parameter is an integer value (not a string) and starts from zero (0).
  - If ”Contact” object with such “index“ does not exist, the method will return "null".
- *method* **AddContact(***phone***):**
  - Return "Contact" object reference.
  - Create new contact or retrieve an existing one with the provided as parameter "phone" number.
- *method* **DelContact(***phone***):**
  - Return no value.
  - Delete existing contact with the provided as parameter "phone" number.
- *method* **SavePrivacy():**
  - Return no value.
  - Save the privacy rules to the data folder on the computer of the web site visitor. Zoiper Web will not save the privacy rules locally, unless this method is used.
  - If the privacy rules exist, they will be loaded upon phone start-up.

Configuration management

- *property* **Version:**
  - Read only.
  - Gets the version string of the currently running Zoiper Web. It can be used to inform the user that there is a newer version and guide him to download and install it.
- *property* **SaveHistory:**
  - Read/Write.

- Governs if Zoiper Web is going to save its history locally upon receiving new history events. The history file is named “History.txt”. It is located in subfolder named “Zoiper Web”, placed in the current user’s application data folder.
- Possible values are: “true”, “false”.

- *property* **SaveContacts:**
  - Read/Write.
  - Governs if Zoiper Web is going to save its contact entries locally upon creation and editing. The contacts file is named “Contacts.xml”. It is located in subfolder named “Zoiper Web”, placed in the current user’s application data folder.
  - Possible values are: “true”, “false”.
  - * You can choose if the contacts created through the Zoiper Web API should be saved locally or not.
- *method* **SaveOptions():**
  - Return no value.
  - Save the Zoiper Web configuration to the application data folder on the computer of the web site visitor.
  - If there are locally stored configurations, they will be loaded on phone startup.
  - * Zoiper Web configurations will not be saved locally, unless this method is used.
- *method* **GetConfig():**
  - Return “Config” object reference.
  - Obtain the reference to the object of class “Config”, which is responsible for the general configuration of the phone.
  - * Only one instance of this class can be obtained and it can be done by this method only.
- *method* **ShowAudioWizard():**
  - Return no value.
  - Shows audio wizard dialog.
- *callback* **OnZoiperReady(***phone***):**
  - This is a callback function that is called by Zoiper Web when the phone has finished loading and it is ready for use.
  - This callback should be used as a main entry point for configuring the phone.
  - Get object reference to the "phone" class provided as parameter.
  - The function will be called before any other callback. Do not use any methods or properties before this callback is called.

Miscellaneous properties and methods

- *property* **MuteMicrophone:**
  - Read/Write.
  - Enable or disable the audio input.
  - Possible values are: “true”, “false”.
- *property* **MuteSpeaker:**
  - Read/Write.
  - Enable or disable the audio output.
  - Possible values are: “true”, “false”.
- method **ShowLog():**
  - Return no value.
  - Show log window containing text information about the actions done by Zoiper Web.
  - If “Config.EnableDebugLog” is enabled, the log will provide and debug information as well.
- *method* **CheckCertificate(***url***):**
  - Return no value.
  - Use to validate the Zoiper Web Certificate from a URL location different from the default one. By default the Zoiper Web Certificate location should be the root folder of the site, where Zoiper Web is installed. The Certificate file is called “activex.certificate”.
  - This method is supposed to be used ONLY in cases, where it is impossible to place the Certificate file in the root folder.
  - The desired URL is specified in the “url” parameter.
- *method* **UseCertificateServer(***ip_address***):**
  - Does not return a value.
  - Use this method to validate the Zoiper Web Certificate from a Zoiper Web certificate server. This is used for local area network (LAN) deployment.
  - Use the “ip_address” parameter to specify the IP address of the certificate server you want to connect to. If a blank string is used Zoiper Web will try to find a certificate server on the LAN automatically using broadcast packets.
  - This method should be called after a call to “Config.SetSIPIAXPorts” method or it will not have effect.
  - * This is only used if you have obtained a license for Zoiper Web certificate server instead of a standalone license.
- *method* **AllowMultipleInstances():**
  - Return no value.
  - Allow more than one instance of Zoiper Web to be run.

- Unless this method is called, every attempt to start new Zoiper Web instance, will result in showing inform message to the user, that there is already a running instance.
- * Zoiper Web cannot run multiple instances in one process. For example: if the Mozilla Firefox web browser is used and you try to open two pages containing Zoiper Web, it will work properly only in the first one. This behavior is will not

be changed by the method **"AllowMultipleInstances()"**. On the other hand you can use Internet Explorer 8, which runs every open tab in a separate process. Thus by using this method you will be allowed to run multiple instances of Zoiper Web (* earlier versions of Internet Explorer will behave like Firefox).

- This method will also allow you to run Zoiper Web from different browsers simultaneously, as they are different processes.

- *method* **ShowDialPad(***show***):**

- Return no value.
- Show or hide the dial pad buttons from Zoiper Web interface according to the "show" parameter.
- The parameter "show" can have the values "true" or "false".

4. "Config" class

The "Config" class encapsulates the Zoiper Web general configuration options. An instance of that class can only be obtained from the "Phone" class by using the "GetConfig" method.

- *property* **PopupMenuOnIncomingCall:**
  - Read/Write.
  - Enable or disable the pop-up menu on incoming new call.
  - Possible values are: "true", "false".
- *property* **PopupOnIncomingCall:**
  - Read/Write.
  - Enable this property to popup Zoiper Web along (with the browser containing it) in front of the other applications when there is a new incoming call or chat message.
  - Possible values are: "true", "false".
- *property* **OnTransferRequest:**
  - Read/Write.
  - Select the behavior of Zoiper Web on transfer request from the remote peer.
  - Possible values are:

    * "accept" - always automatically accept the requests; * "reject" - always automatically reject the requests; * "ask" - ask the user what action to be taken.

- *property* **UseEchoCancelation:**

- Read/Write.
- Enable or disable echo cancellation.

- Possible values are: “true”, “false”.

- *property* **DebugLogPath:**
  - Read/Write.
  - Set the path, where the debug log file should be stored, if the "EnableDebugLog" is set to “true”.
- *property* **EnableDebugLog:**
  - Read/Write.
  - Enable or disable the debug log of the Zoiper Web.
  - Possible values are: “true”, “false”.
- *property* **EnableDebugLargeDump:**
  - Read/Write.
  - Enable or disable the generation of a large crash dump file that includes memory state information along with the call stack and register information for debugging purposes.
  - * Files generated with this option enabled become much larger (200MB) than the small crash dump files (1MB).
  - Possible values are: “true”, “false”.
- *property* **Restrictions:**
  - Read/Write
  - Apply restrictions to the Zoiper Web interface. The restriction entries are comma separated values that can be used in any combination. The valid values for the restrictions are:
    * options_sip_accounts – restrict the access to the entire SIP account options.
    * options_iax_accounts – restrict the access to the entire IAX account options.
    * options_sip_accounts_overview – restrict the access to the overview SIP account options.
    * options_iax_accounts_overview – restrict the access to the overview IAX account options.
    * options_sip_new_account – restrict the creation of new SIP accounts.
    * options_iax_new_account – restrict the creation of new IAX accounts.
    * options_audio_general – restrict the access to the audio general options.
    * options_audio_devices – restrict the access to the audio devices options.

* options_audio_codecs – restrict the access to the audio codec options.
* options_general – restrict the access to the general options.
* options_call_events – restrict the access to the general call events options.
* options_integration – restrict the access to the general integration options (* not supported in all versions).
* options_provision – restrict the access to the provisioning options (* not supported in all versions).
* options_sip – restrict the access to the SIP protocol options.
* options_iax – restrict the access to the IAX protocol options.
* options_rtp – restrict the access to the RTP protocol options.
* options_stun – restrict the access to the STUN protocol options.
* options_diagnostics – restrict the diagnostic options.
* options_network – restrict the access to the network options.
* options_fax – restrict the access to the fax options.
* options_video – restrict the access to the video options (* not supported in all versions).
* options_chat – restrict the access to the chat options (* not supported in all versions).
* options_contact_server – restrict the access to the contact server options (* not supported in all versions).
* options_show_advanced – restrict the access to the “Show advanced options” check box in the options window and disabling the access to all the advanced options.
* options – restrict the showing of the options window.
* dial – restrict the ability to dial.
* hangup – restrict the ability to hang up.
* transfer – restrict the ability to transfer.
* hold – restrict the ability to hold.
* record – restrict the ability to record calls (* not supported in all versions).
* conference – restrict the ability to start conference call (* not supported in all versions).
* auto_answer – restrict the ability to auto answer calls (* not supported in all versions).

  - fax – restrict the ability to send faxes (* not supported in all versions).
  - video - restrict the ability to start video calls (* not supported in all versions).
  - history – restrict the ability to view the history.
  - contacts – restrict the ability to view the address book / contacts.
  - dialpad – restrict the access to the dial pad.
  - chat – restrict the access to the chat window (* not supported in all versions).
  - chat_history - restrict the access to the chat history (* not supported in all versions).
  - save_contact – restrict the access to the save contact button (* not supported in all versions).
  - find_contact – restrict the access to the find contact button (* not supported in all versions).
  - send_message – restrict the access to the send message button (* not supported in all versions).
  - account_register – restrict the access to the account register button.
  - account_unregister – restrict the access to the account unregister button.
  - wizard_audio – restrict the access to the audio wizard button.

- property **ServerSideAutoAnswer:**
  - Read/Write.
  - Enable or disable the ability of the phone to favor SIP “Call-Info” auto answer headers.
  - Possible values are: “true”, “false”.
  - * Property not supported in all versions.
- property **AutoAnswer:**
  - Read/Write.
  - Enable or disable the auto answer feature of the phone.
  - Possible values are: “true”, “false”.
  - * Property not supported in all versions.
- property **AutoAnswerSeconds:**
  - Read/Write.
  - Set the delay before the call is auto answered.
  - The value set is in seconds.
  - * The property must be set before the “AutoAnswer” one is set to “true”.

- * Property not supported in all versions.

- property **AutoAnswerPlaySound:**
  - Read/Write.
  - Enable or disable the sound played upon auto answering.
  - Possible values are: “true”, “false”.
  - * Property not supported in all versions.
- property **AudioInputVolume:**
  - Read/Write.
  - Control the input (microphone) volume.
  - Possible values are between “1” and “100”.
- property **AudioOutputVolume:** - Read/Write
  - Control the output (speaker/headphones) volume.
  - Possible values are between “1” and “100”.
- property **MediaDSCP:**
  - Read/Write.
  - Control the DSCP tagging for the outbound media streams.
  - Possible values are: *"NONE"*, *"CS0"*, *"CS1"*, *"CS2"*, *"CS3"*, *"CS4"*, *"CS5"*, *"CS6"*, *"CS7"*, *"AF11"*, *"AF12"*, *"AF13"*, *"AF21"*, *"AF22"*, *"AF23"*, *"AF31"*, *"AF32"*, *"AF33"*, *"AF41"*, *"AF42"*, *"AF43"* and *"EF"*.
  - * Windows might not favor this setting, unless the following registry key is not set to “0”:
    HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters \ DisableUserTOSSetting
- property **RTPPortBase:**
  - Read/Write.
  - Control the port base used to create RTP sessions. It should have an even value. By RFC recommendations even ports are used for RTP connections and odd ports are used for RTCP connections. Both connections are always present. If by mistake an odd value is configured, the actual value used for RTP port base will be automatically set to the next even value.
  - Values between “1” and “65500” are accepted, but it is strongly recommended to use ports above 1024. Otherwise there is a possibility for port collisions.
- property **RTPUseRandomPort:**
  - Read/Write.
  - Enable or disable the usage of random value for RTP port base. If enabled, the port base will be randomly selected from ports between 32000 and 65500.
  - Possible values are: “true”, “false”.
- property **SIPTLSCertificateFile:**

- Read/Write.
- Specify file path for additional TLS certificate.
- The certificate file should be in PEM format and it is allowed to hold several certificates
- * Property not supported in all versions.

- property **Forwarding:**
  - Read/Write.
  - Enable or disable the automatic call forwarding.
  - The property will work, only if correct value is set to the "**ForwardingExtension**" property.
  - Possible values are: "true", "false".
  - * Property not supported in all versions.
- property **ForwardingSeconds:**
  - Read/Write.
  - Set the time in seconds to wait before the call to be forwarded. It will be forwarded to the number set to the "**ForwardingExtension**" property.
  - Possible values are between "0" and "60000".
  - * Property not supported in all versions.
- property **ForwardingExtension:**
  - Read/Write.
  - Set the phone number to which the call should be forwarded.
  - * Property not supported in all versions.
- property **RecordCalls;** - Read/Write.
  - Enable or disable the automatic call recording.
  - When enabled, all calls will be recorded. The recordings will be stored in the specified by the "**RecordPath**" property path.
  - Records will be named with the prefix "recorded_conversation_" plus the date and time of the call.
  - Records will be stored in ".wav" file format.
  - Possible values are: "true", "false".
  - * Property not supported in all versions.
- property **RecordPath:**
  - Read/Write.
  - Set the path to store the call recordings.
  - * Property not supported in all versions.
- property **RingWhenTalking:**

- Read/Write.
- Enable or disable the ringing sound in case of incoming call while there is already another established call.
- Possible values are: “true”, “false”.

- property **ChatPlaySound:**
  - Read/Write.
  - Enable or disable the playing of sound in case of incoming chat message.
  - Possible values are: “true”, “false”.
  - * Property not supported in all versions.
- property **ChatBlinkWindow:**
  - Read/Write.
  - Enable or disable the blinking of the chat window on the task bar upon receiving a new chat message.
  - Possible values are: “true”, “false”.
  - * Property not supported in all versions.
- property **ChatShowDialog:**
  - Read/Write.
  - Enable or disable the showing of a chat dialog upon receiving a new chat message.
  - The property is used when the chat is handled entirely by using the API.
  - Possible values are: “true”, “false”.
  - * Property not supported in all versions.
- *property* **DoNotShowError:**
  - Read/Write.
  - Enable or disable the error messages shown by the phone.
  - Possible values are: “true”, “false”.
- *property* **PCSpeakerRing:**
  - Read/Write.
  - Enables or disables the ringing using the PC speaker.
  - Possible values are: “true”, “false”.
  - * Property not supported in all versions.
- *property* **DisableDTMFSounds:**
  - Read/Write.
  - Enables or disables the sounds played on the output device when a DTMF signal is sent over the line.
  - Possible values are: “true”, “false”.

- * Note that this doesn't affect the actual DTMF signal sending.

- *property* **MuteEarlyMedia:**
  - Read/Write.
  - Enables or disables the automatic muting of early media audio received from the the PBX on outgoing calls if available.
  - Possible values are: “true”, “false”.
  - * Property not supported in all versions.
- *property* **PhoneInfoText:**
  - Read/Write.
  - Sets the default text displayed in the “phone to dial” field when there is nothing entered by the user.
  - The default value is “Find a contact”.
- method **NumberOfCallsLimit(***numberOfCalls***):**
  - Return no value.
  - Set the maximum amount of simultaneous incoming/outgoing calls.
  - Possible values are between “0” and “6”
  - When the maximum amount of calls reached, the phone will refuse to dial more calls and will reject any new incoming calls.
- method **SetSIPIAXPorts(***sipPort, iaxPort***):**
  - Return no value.
  - Set the listen ports for SIP and IAX connections to provided as parameters “sipPort” and “iaxPort”.
  - Possible values for the ports are between “1” and “65500”.
  - * The usage of this method to change ports will cause the phone to stop all of its current calls and restart internally. That is why, it is recommended to use this method only at the beginning of the setup stage. In other words at the beginning of the “OnZoiperReady” callback implementation.

5. The "Account" class

The "Account" class represents a Zoiper Web connection to a given VoIP service, its properties and basic functions. Zoiper Web can't make calls without an "Account" class instance. Note that some VoIP services will allow making calls without registering first while others don't. It depends on the service. Changing the properties will take effect only after the "Apply" method is called.

- *property* **Name:**
  - Read only.

- Use to read the account name. It acts as unique identifier for the account.

- *property* **Domain**:
  - Write only.
  - Set the SIP domain and port for the account. To set the port use the form <*domain*>:<*port*>
- *property* **OutboundProxy**:
  - Write only.
  - Set the SIP outbound proxy address and port for the account. To set the port use the form <*domain*>:<*port*>
- *property* **Host**:
  - Write only.
  - Set the IAX2 or XMPP host and port for the account. To set the port use the form <*domain*>:<*port*>
- *property* **Context:**
  - Write only.
  - Set the IAX2 context for the account.
- *property* **CalledID:**
  - Write only.
  - Sets the caller identification name for the account. It applies to SIP, IAX2 and XMPP accounts.
- *property* **Phone:**
  - Write only.
  - Sets the caller identification number for the account. It applies to IAX2 accounts.
- *property* **UserName:**
  - Write only.
  - Sets the user name used by VoIP service to identify the account.
- *property* **JabberID:**
  - Write only.
  - Set the user identification for XMPP accounts.
  - It is used along with the **"Host"**, **"Password"** and **"CallerID"** properties to define the XMPP connection parameters.
  - * Property not supported in all versions.
- *property* **AuthenticationUserName:**
  - Write only.
  - Sets the username used by the VoIP service for the account authentication.

- * In most cases “**UserName**” and “**AuthenticationUserName**” are one and the same parameter. In these cases you can just set the “**UserName**” property.

- *property* **Password:**
  - Write only.
  - Sets the password used by the VoIP service to authenticate the account.
- *property* **EncryptedPassword:**
  - Write only.
  - Stores the password used to authenticate the account with the voip service in encrypted form.
  - Can be used only after acquiring a Zoiper Web certificate that contains a password encryption key. Such a certificate and a password encryption key can be acquired by contacting sales@zoiper.com.
  - The password should be encrypted using AES-256 in CBC mode with initialization vector of zeroes and the password encryption key obtained.
  - Every Zoiper Web certificate has its own unique password encryption key.
- *property* **STUNHost:**
  - Write only.
  - Set the host address of the SIP STUN server for the account.
  - Set an empty string if you want to disable the STUN usage for the account.
  - The STUN usage is disabled by default.
- *property* **STUNPort:**
  - Write only.
  - Set the port for the SIP STUN server.
  - Possible values are between “1” and “65500”.
- *property* **RegistrationExpiry:**
  - Write only.
  - Set the registration expiration time for the account in seconds.
  - For SIP accounts the default value is “3600” seconds.
  - For IAX2 accounts it is “60” seconds.

-
  * The value set by this property will be suggested by Zoiper Web to the VoIP service, but it can be overridden by it.

- *property* **DisableRingbackTones:**
  - Write only.
  - Enable or disable the playback of ringback tones, while Zoiper Web is waiting for the remote peer to answer.
  - Possible values are: “true”, “false”.
- *property* **UseRPort:**
  - Write only.
  - Enable or disable the use of RPort for the account.
  - Possible values are: “true”, “false”.
- *property* **UseRPortMedia:**
  - Write only.
  - Enable or disable the use of media RPort for the account - Possible values are: “true”, “false” ▪ *property* **ForceRFC3264:**
  - Write only.
  - Enable or disable the forced use of RFC-3264 for the account.
  - Possible values are: “true”, “false”.
- *property* **PresenceModel:**
  - Write only.
  - Set the presence model used by the account.
  - Possible values are:
    * “server_based”
    * “peer-to-peer” ▪ *property* **DTMFType:**
  - Write only.
  - Set the DTMF type to be used by the account.
  - Possible values are:
    * “media_outband” - RFC-2833 is going to be used for the SIP accounts. Outofband DTMF signaling will be used for the IAX2 accounts;
    * “signalling_outband”- SIP INFO will be used for SIP accounts. Out-of-band DTMF signaling will be used for the IAX2 accounts;

-
  - * “media_inband” - Inband DTMF signaling will be used for both SIP and IAX2 accounts;
  - * “disabled” - Disable the sending of DTMF signals.

- *property* **SIPTransportType:**
  - Write only.
  - Set the transport type used by the SIP protocol.

    Possible values are:
    - * “udp” - UDP transport protocol will be used. This is the default value;
    - * “tcp”- TCP transport protocol will be used;
    - * “tls” - TLS protocol over TCP protocol will be used; TLS protocol is used to encrypt the SIP data using the certificates already installed on the machine. Additional certificates can be added by using the “**Config.SIPTLSCertificateFile**“ property.
  - * Property not supported in all versions.
- *property* **RTPEncryption:**
  - Write only.
  - Set the RTP encryption method.
  - Possible values are:
    - * “none” – The RTP will not be encrypted. This is the default value;
    - * “sdes”- SDES encryption will be applied to the RTP streams. Note that this encryption method can be used only with the “**SIPTransportType**” property set to “tls” because the encryption key should be transferred between both sides in a secure way.
  - * Property not supported in all versions.
- *property* **UseLegacyTLS:**
  - Write only.
  - Set the TLS mode used to connect to the server.
  - The property is used for XMPP accounts.
  - Possible values are: “true”, “false”.
  - * Property not supported in all versions.
- *property* **UseRoster**:
  - OBSOLETE. No longer used in recent versions of Zoiper Web - Write only.
  - Set the contact roster to be downloaded from the server and new contacts to be added (if there are any).

-
- The property is used for XMPP accounts.
- Possible values are: “true”, “false”.
- * Property not supported in all versions.

- *property* **CreateAccount:**
  - Write only.
  - Set the account to be created automatically on the server.
  - The property is used for XMPP accounts.
  - Possible values are: “true”, “false”.
  - * In order to work properly, the XMPP server should have this feature enabled, otherwise the registration to the server will be rejected.
    * Property not supported in all versions.
- *property* **Status**:
  - Read only.
  - Gets the account registration status.

-

  Possible values are: “NotRegistered”, "Registering", "Registered" or “WaitForNetworkDiscovery”. The last one is returned when the account is used along with STUN server and is waiting for the STUN network type discovery.

- *property* **SubscribeForMWI**:
  - Write only.
  - Set when the account is going to subscribe for its message waiting indication (MWI).
  - Can have the following values :
    * “disabled” - no subscription for MWI will be made;
    * “before”- the subscription for MWI will be made before the account registration;
    * “after” - the subscription for MWI will be made after the account registration.
    * “both” - two subscriptions for MWI will be be made. One before and one after the account registration.
  - This property must be set before invoking “Apply” and “Register” methods.
- *method* **Apply():**
  - Return no value.
  - Apply any changes made to the “Account” object properties.
- *method* **Register():**
  - Return no value.
  - Use to try to register the account with the VoIP service specified in the properties.
  - * This method must be called after the "**Apply**" one.
  - The result of the register attempt will be returned in either "**OnZoiperAccountRegister**" or "**OnZoiperAccountRegisterFail**" callbacks.
- *method* Unregister()
  - Return no value.
  - Use to try to register the account with the VoIP service specified in the properties.
  - * This method must be called after the "**Apply**" one.
  - The result of the register attempt will be returned in either "**OnZoiperAccountRegister**" or "**OnZoiperAccountUnregisterFail**" callbacks.

-

- *method* **AddCodec(***codec***):**
  - Return no value.
  - Add a "codec" that can be used by the account.

-

The "codec" parameter can have the following values:

* "GSM", "a-law", "u-law", "Speex", "iLBC 30" and "iLBC 20".

- The first call to this method will enable the custom codec list for the account.

If the method is not called, the account will use the global codec list, which by default has all the available codecs enabled.

- *method* **ClearCodecs():**
  - Return no value.
  - Disable the custom codec list for the account.
- *method* **SipHeaderDump(***enable***):**
  - Return no value.
  - Enable or disable the call of "**OnZoiperCallSipHeader**" callback.
  - It applies to the calls associated with the account.
  - By default this option is disabled.
  - The parameter "enable" can take "true" or "false" values.
  - * The method should be called after the call to the "**Apply**" method.
- *method* **SipHeaderAdd(***name, value***)**:
  - Return no value.
  - Add a new header value for the calls associated with the account.
  - The parameter "name" takes the name of the desired header and the parameter "value" the desired value for that header.
  - The method can be used for multiple calls of one and the same header "name". The result will be multiple headers with one and the same name, but with different values.
  - If a header has to be changed, it is essential to first call the "**SipHeaderClear**" method. After that you can call the "**SipHeaderAdd**" one.
  - * The method should be called after the call to the "**Apply**" method.
- *method* **SipHeaderClear(***name***):**
  - Return no value
  - Use to clear all of the custom headers added by using the "**SipHeaderAdd**" method.
  - The parameter "name" takes the name of header, which value has to be cleared.
  - * The method should be called after the call to the "**Apply**" method.
- *callback* **OnZoiperAccountRegister(***account***)**:
  - Callback function called by Zoiper Web upon a successful registration of an account.

-
-

- Object reference to the "Account" provided as parameter.

- *callback* **OnZoiperAccountUnregister(***account***):**

  - Callback function called by Zoiper Web upon a successful unregistration of an account. Object reference to the "Account" provided as parameter.

- *callback* **OnZoiperAccountRegisterFail(***account***):**

  - Callback function called by Zoiper Web upon a failed registration or unregistration of an "Account".

  Object reference to the "Account" provided as parameter.

6. The "Call" class

The "Call" class represents a call instance in Zoiper Web, its properties and methods. The "Call" is always done using an "Account" but there are cases of incoming calls which can't be matched to an existing "Account". In this case the "Account" property holds a temporary "Account" created just for that "Call".

- *property* **Phone:**
  - Read only.
  - The property will get the caller identification name in case of incoming call - The property will get the dialed number in case of outgoing call.
- *property* **DNID:**
  - Read only.
  - The property will get the Dialed Number Identifier (the number that the other end has dialed to make the call) in case of incoming calls.
- *property* **Account:**
  - Read only.
  - The property will get a reference to the "Account" object used for the call.
- *property* **Contact:**
  - Read only.
  - The property will get a reference to the "Contact" object used for the call.
  - When a phone is dialed, Zoiper Web tries to math the phone number to a contact from the phonebook. If there is one the Zoiper Web will assign the call to the

- found contact. If there is no matching contact, a temporary one will be created for the call.

- *property* **Duration:**
  - Read only.
  - Return the call duration in seconds.

    In case of incoming call that is not picked up yet, it will return the duration of the ringing. When the call is answered the property is reset to zero.
- *property* **InConference:**
  - Read/Write.
  - Put the call in and out of conference.
  - Zoiper Web supports only one conference at a given time.
  - Possible values are: “true”, “false”.
  - If the property is set to “true”, the call will join the conference. If it is “false” the call will be put out of the conference.

-

Read the property to get information whether the call is in conference or not.

- * Property not supported in all versions.

- *property* **IsHold:**
  - Read only.
  - Indicates if the call is on hold.
  - Possible values are: “true”, “false”.
- *property* **IsRemoteHold:**
  - Read only.
  - Indicates if the call is put on hold from the remote end.
  - Possible values are: “true”, “false”.
- *property* **IsIncoming:**
  - Read only.
  - Indicates if the call is incoming or outgoing.
  - Possible values are: “true”, “false”.
- *property* **IsRecording:**
  - Read only.
  - Indicates if the call is being recorded.
  - Possible values are: “true”, “false”.
- *property* **IsTransferring:**
  - Read only.
  - Indicates if the call is in transferring state.
  - Possible values are: “true”, “false”.
- *property* **IsRinging:**
  - Read only.
  - Indicates if the call is ringing.
  - Possible values are: “true”, “false”.
- *method* **Hang():**
  - Return no value.
  - Hang up the call.
  - If you call this method, no other references to this "Call" object should be made as it is destroyed.
- *method* **Hold():**

- Return no value.

- Put the call on hold. If the call is the active call in the "Phone" object, the active call will be set to "null".

method **UnHold():**

- Return no value.
- Unhold the call. If there is another "Call" object in the "Phone" object it will be put on hold automatically and the new call will become the active one.
- The method works in the same way as the "**SetActive**" one.

- *method* **SetActive():**
  - Return no value.
  - Set the specified call to be the active one thus holding any other "Call" that might be active at this moment. It will also unhold this "Call" and will result in visual line change. This method does the same as the "**UnHold**" method.
- *method* **Accept():**
  - Return no value.
  - Accept incoming call.
- *method* **Reject():**
  - Return no value.
  - Reject incoming call.
- *method* **Transfer(***number***):**
  - Return no value.
  - Make an unattended transfer to the "number" provided as parameter.
  - * You cannot transfer a call that is using certain VoIP service towards another VoIP service.
- *method* **TransferAttended(***call***):**
  - The method returns "true" if the transfer is successful and "false" if it is not.
  - Make an attended transfer to the "call" provided as parameter.
  - The parameter "call" has to be another "Call" object.
  - In order to work properly, the current "Call" object has to be active (unholded) and the "Called" object should be on hold.
  - * You cannot transfer a call that is using certain VoIP service towards another VoIP service.
  - * Method not supported in all versions.
- *method* **SendDTMF (***dtmf_sequence***):**

-
- Return no value.
- Send a DTMF sequence.
- The parameter “dtmf_sequence” could contain a single character or a sequence of characters to be converted into DTMF signals according to the property **“DTMFType”** set for the account, used for the call.
- The valid characters to be used with this method are:

  * “0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “a”, “b”, “c”, “d”, “*” and “#”.
- * The usage of the DTMF type **“signalling_outband”** could result in slower sending of the DTMF signals, because it requires a confirmation to be received from the remote end.

- *method* **StartRecording():**
  - The method returns “true” if the recording is started successfully and “false” if it is not.
  - Starts to record the call to the destination specified in the configuration objects “RecordPath” property.
  - The recording file name could be obtained in the “OnZoiperCallStartRecording” callback.
  - * Method not supported in all versions.
- *callback* **OnZoiperCallFail(***call***):**
  - Callback function called by Zoiper Web, when the call is failed for some reason.
  - Return object reference to the "Call" provided as parameter.
- *callback* **OnZoiperCallRing(***call***):**
  - Callback function called by Zoiper Web, when the call has just finished the dialing and the remote peer has started to ring.
  - Return object reference to the "Call" provided as parameter.
- *callback* **OnZoiperCallHang(***call***):**
  - Callback function called by Zoiper Web, when the Call is hung up.
  - Return object reference to the "Call" provided as parameter.
-

- *callback* **OnZoiperCallHold(***call***):**
  - Callback function called by Zoiper Web, when the remote peer puts the call on hold.
  - Return object reference to the "Call" provided parameter.
- *callback* **OnZoiperCallUnhold(***call***):**
  - Callback function called by Zoiper Web, when the remote peer resumes (unholds) the call.

    Return object reference to the "Call" provided as parameter.
- *callback* **OnZoiperCallAccept(***call***):**
  - Callback function called by Zoiper Web, when the remote peer accepts the call.
  - Return object reference to the "Call" provided as parameter.
- *callback* **OnZoiperCallReject(***call***):**
  - Callback function called by Zoiper Web, when the remote peer rejects the call.
  - Return object reference to the "Call" provided as parameter.
- *callback* **OnZoiperCallIncoming(***call***):**
  - Callback function called by Zoiper Web, when there is a new incoming call.
  - Return object reference to the "Call" provided as parameter. *callback* **OnZoiperCallRecvDTMF(***call, dtmf***):**
  - Callback function called on incoming DTMF.
  - Return object reference to the "Call" provided as the first parameter.
  - The second parameter that is provided is the DTMF code.
  - * The only supported DTMF signaling type for the callback is “out-of-band”.
- *callback* **OnZoiperCallSipHeader(***call, headers***):**
  - Callback function called, when a “SIP” call provides its headers.
  - Return object reference to the "Call" provided as the first parameter.
  - The second parameter that is provided is a reference to the “Header” object containing the “SIP” header for the call.
- *callback* **OnZoiperTransferSuccess(***call***):**
  - Callback function called, when a blind transfer request has succeeded.
  - Return object reference to the "Call" provided as parameter.
- *callback* **OnZoiperTransferFailed(***call***):**
  - Callback function called, when a blind transfer request has failed.
  - Return object reference to the "Call" provided as parameter.
- *callback* **OnZoiperCallStartRecording(***call,filename***):**
  - Callback function called, when the phone has started to record a call.

-
- Return object reference to the "Call" provided as the first parameter.
- The second parameter that is provided contains the file name of the recording.
- * Callback not supported in all versions.

7. The "Contact" class

The "Contact" class represents a single contact in the Zoiper Web address book. It is used to store information about the "Contact" properties.

- *property* **Name:**
  - Read only.
  - Get the contact display name. If the display name is not set, the result will be formed by the first name, middle name and last name, in that order.
- *property* **Phone:**
  - Read only.
  - Obtain the contact's primary phone number.
- *property* **Account:**
  - Read/Write.
  - Set or get the "Account" object name associated with the contact.
- *property* **PresenceAccount:**
  - Read/Write.
  - Use to specify the "Account" object name to be used to obtain the presence and the chat information for the contact.
- *property* **Display:** - Read/Write
  - Sets the contact's display name.
- *property* **FirstName:** - Read/Write
  - Sets the contact's first name.
- *property* **LastName:** - Read/Write
  - Sets the contact's last name.
-

- *property* **MiddleName:** - Read/Write - Sets the contact's middle name.
- *property* **Country:** - Read/Write - Sets the contact's country name.
- *property* **City:**
  - Read/Write
  - Sets the contact's city name.
- *property* **WorkPhone:** - Read/Write
  - Sets the contact's work phone number.
- *property* **HomePhone:** - Read/Write
  - Sets the contact's home phone number.
- *property* **CellPhone:** - Read/Write

  Sets the contact's cell phone number.
- *property* **FaxNumber:** - Read/Write
  - Sets the contact's fax phone number.
- *property* **SaveContact:** - Read/Write
  - Enable the contact to be stored locally.
  - * If the "Phone" class property "**SaveContacts**" is set to "false", the contact will not be saved regardless what is set to this property.
- *property* **PresenceID:**
  - Read/Write.
  - Set the presence and chat identification string of the "contact. - Used for the SIP SIMPLE and XMPP protocols to identify the contact. *property* **Status:**
  - Read only.
  - Get the current presence status of the contact.
  - Possible values are:
    * "Offline", "Invisible", "Online", "Away", "Busy", "On the phone", "Out to lunch" and "Be right back".
  - * The "**PresenceAccount**" property should be set and the "Account" object specified in this property should be registered.
- *method* **Apply():**
  - Return no value.
  - Apply any changes made to the "Contact" object properties.
- *method* **SendMessage(***message, show***):**
  - Return no value.

-
- Try to send a message to a contact.
- The first parameter is the message itself.
- The second parameter can be “true” or “false” depending on whether to enable or disable a pop-up chat window upon the message sending.
- The result of the operation will be presented either in the “**OnZoiperContactMessageSent**” or in the “**OnZoiperContactMessageFailed**” callbacks.
- * Method not supported in all versions.

- *callback* **OnZoiperContactStatus(***contact, status***):**

-

-

-

Callback function called by Zoiper Web, when the contact status is changed.

Return reference to the "Contact" object provided as first parameter.

- The second parameter represents the status of the "Contact" and can be one of the following values: “Offline”, “Invisible”, “Online”, “Away”, “Busy”, “On the phone”, “Out to lunch” and “Be right back”.

- *callback* **OnZoiperContactCreate(***contact***):**
  - Callback function called, when a new contact is created.
  - Return reference to the newly created “Contact” object provided as parameter.
- *callback* **OnZoiperContactDelete(***contact***):**
  - Callback function called, when a contact is deleted.
  - Return reference to the deleted “Contact” object provided as parameter.
  - * The reference is obsolete and should not be used.
- *callback* **OnZoiperContactChange(***contact***):**
  - Callback function called when a contact is edited.
  - Return reference to the “Contact” object provided as parameter.
- *callback* **OnZoiperContactMessageReceived(***contact, message***):**
  - Callback function called, when a chat message is received.
  - The first parameter is the contact from which the message is coming.
  - The second parameter contains the message itself.
  - * Callback not supported in all versions.
- *callback* **OnZoiperContactMessageSent(***contact, message***):**

  Callback function called, when a message is sent successfully.

  The first parameter holds a reference to the “Contact” object used for the message sending.
  - The second parameter holds the message itself.
  - * Callback not supported in all versions.
- *callback* **OnZoiperContactMessageFailed(***contact, message, cause***):**
  - Callback function called, when the sending of a message fails.
  - The first parameter holds a reference to the “Contact” object used for the message sending.
  - The second parameter holds the message itself.

-
-
- The third parameter holds the error code that caused the failure.
- * Callback not supported in all versions.

8. The "Header" class

The "Header" class represents a collection of headers used by the “SIP” protocol on incoming call. It is allowed to have one header with more than one value. This way you can make the “Header” class to return different collection of properties for each header.

- *property* **Count:**
  - Read only.
  - Return integer value.
  - Count the headers represent in the “Header” object.
- *method* **Entry(***index***):**
  - Return reference to the “Property” object.
  - The parameter “index” could be an integer value and should be less than the value of the property “Count”.
  - Could be a string containing the name of the header. If such header is available a reference to the “Property” object will be returned. Otherwise the method will return “null”.

9. The "Property" class

The "Property" class represents a collection of strings and integer.

- *property* **Name:**

  Read only.

  Holds the name of the “Property” contained in this object.
- *property* **Count:**
  - Read only.
  - Integer property.

-

-

- Count the strings contained in the “Property” object.

- *method* **Value(***index***):**

  - Return string value corresponding to the parameter “index”

  - Should be an integer value

  - It has to be less than the value of the property “Count”.

Example

Here is an example on how to use Zoiper Web on the supported browsers. The explanations on the example are inline as comments. The comment references the next line or block of lines after it.

```
<html>
<head>
	<title>Zoiper Web Example</title>
</head>

<!-- We give the body a function to be called upon page unload for finalization of Zoiper Web
-->
<body onunload="Quit()">
<script type="text/javascript">
	var Zoiper; /* The variable "Zoiper" is used to keep reference about the Zoiper Web
instance. It is going to hold the "Phone" object */
	var ActiveCall; /* The variable "ActiveCall" is used to keep a reference to the current active
"Call" object */

	/*The function GetValue is a helper function used to get the value of a document element
by its ID */
	function GetValue(name)
	{
		return document.getElementById(name).value;
	}

	/* The function "Quit" is used to stop Zoiper Web, we will use it to delete the items
we have created during the example */		function Quit()
	{
		Status("Quit() called");
if (Zoiper != null)
		{
			Zoiper.DelContact("web demo");
			Zoiper.DelAccount("Sample");
}

		/* On Firefox new instances of the plugin are created before old ones are destroyed
Because of this we must manually destroy the existing plugin */
document.getElementById('ZoiperA').innerHTML = "";
	}

	/* The function "Hang"is a helper function and is used to hangup the "ActiveCall" that we
keep reference of */
	function Hang()
	{
		if (null != ActiveCall)
		{
			ActiveCall.Hang();
			ActiveCall = null;
		}
	}

	/* The function "Dial" is used to dial a number entered by the user in the input field with
ID="number". It also assigns the newly created "Call" object to the "ActiveCall" variable */
function Dial()
	{
		ActiveCall = Zoiper.Dial(GetValue("number"));
}

	/* The function "Hold" is used to put the current active call on hold */		function
Hold()
	{
		if (null != ActiveCall)
		{
			ActiveCall.Hold();
}
```

```
    }

    /* The function "SendDTMFSequence" sends a sequence of dtmf digits entered in the
input area named "dtmfsequence" */
    function SendDTMFSequence()
    {
        if (null != ActiveCall)
        {
            ActiveCall.SendDTMF(GetValue("dtmfsequence"));
        }
    }

    /* The function "ShowAudioWizard" is used to show audio wizard dialog */
function ShowAudioWizard()
    {
        if (null != Zoiper)
        {
            Zoiper.ShowAudioWizard();
        }
}

    /* The function "ShowLog" is used to show log dialog */
function ShowLog()
    {
        if (null != Zoiper)
        {
            Zoiper.ShowLog();
        }
    }

    /* The function "MuteSpeakerToggle" is used to enable and disable audio output */
function MuteSpeakerToggle()
    {
        if (Zoiper.MuteSpeaker == "true")
        {
            Zoiper.MuteSpeaker = "false";
        } else
        {
            Zoiper.MuteSpeaker = "true";
        }
    }

    /* The function "MuteMicToggle" is used to enable and disable audio input */
function MuteMicToggle()
    {
        if (Zoiper.MuteMicrophone == "true")
        {
            Zoiper.MuteMicrophone = "false";
        } else
        {
            Zoiper.MuteMicrophone = "true";
        }
    }

    /* The function "Login" is used to login the user to the Zoiper Service. It uses the
username and password provided by the user in the input fields "user" and "pass" */
function Login()
    {
        var user = GetValue("user");
var pass = GetValue("pass");
```

```
            Zoiper.Login(user,pass);
}

    /* The function "Logout" is used to logout the user from the Zoiper Service. */
```

```
	function Logout()
	{
		Zoiper.Logout();
	}

	/* The function "Status" is used to show a status text in the element with ID="Status". It is used as a log function to show the state of the phone and what events are triggered */
function Status(text)
	{
		var node = document.getElementById("thelog");
node.value += text + "\n";
	}

	/* The function "OnZoiperReady" is the entry point for Zoiper Web usage. It is called by Zoiper Web when it is ready for use. It provides a reference to its "Phone" object which we assign to the "Zoiper" variable. We use it to make the initial setup */	function OnZoiperReady(phone)
	{
		document.getElementById("thelog").value = ""; /* We clear the log input box */
Zoiper = phone; /* We put the Zoiper Web instance reference to the global variable "Zoiper" */
		Zoiper.AllowMultipleInstances(); /* Here we allow other Zoiper Web instances in different processes to be loaded if necessary */
		Status("Version : " + Zoiper.Version); /* We print the version of the Zoiper Web instance we are running */

		var Config = Zoiper.GetConfig(); /* Here we get the "Config" object instance and put it in the variable named Config */

		Config.SetSIPIAXPorts("4566","5061"); /* Here we change the SIP and IAX listening ports */
		Config.NumberOfCallsLimit("2"); /* Here we restrict the number of simultaneous calls allowed to 2 */

		/* Here we set some general configuration properties. We disable the popup menu on incoming calls  and we set the debug log to be  written to  “D:\” and enable it */
		Config.PopupMenuOnIncomingCall = "false";
		Config.DebugLogPath = "d:\\";
		Config.EnableDebugLog = "true";
		Config.RingWhenTalking = "false"; /* Here we setup Zoiper Web not to ring when the user is already on the phone */
		Account = Zoiper.AddAccount("Sample", "sip"); /* Here we create or get an existing "Account" object called "Sample". The "Account" is going to use the SIP protocol */

		/* Here we set the "Account" properties */
		Account.Domain   = "10.2.1.9:6060";
		Account.CallerID = "666";
		Account.UserName = "666";
Account.Password = "666";

		/* Here we enable the STUN usage */
		Account.STUNHost = "stun.zoiper.com";
Account.STUNPort = "3478";

		/* Here we enable the custom codec list and select only the GSM codec */
Account.AddCodec("GSM");

		/* Here we set the “Account” to use inband DTMF signals */ Account.DTMFType = "media_inband";

		/* Here we apply the so far set properties and register the "Account" */
Account.Apply();
```

```
Account.Register();
```

```
		Account.SipHeaderDump("true"); /* Here we enable the SIP header dump */

		/* Here we set a custom header named "testheader" to be send in the calls that are using this account. First we set it to "zoiperweb" then we clear it and then add two headers with the same name set to "value1" and "value2" */
		Account.SipHeaderAdd("testheader","zoiperweb");
		Account.SipHeaderClear("testheader");
		Account.SipHeaderAdd("testheader","value1");
		Account.SipHeaderAdd("testheader","value2");

		Zoiper.UseAccount("Sample"); /* Here we select the "Sample" "Account" to be the active one */

		/* Below we create a new "Contact" object with primary number "web demo" and set its properties */
		var Contact		= Zoiper.AddContact("web demo");
		Contact.Account	= "Sample";
		Contact.Display	= "web demo display";
		Contact.FirstName	= "John";
		Contact.MiddleName	= "F.";
		Contact.LastName	= "Doe";
		Contact.Country	= "Alabama";
		Contact.City		= "Huntsville";
Contact.WorkPhone	= "work";
		Contact.HomePhone	= "home";
		Contact.CellPhone	= "cell";
		Contact.FaxNumber	= "fax";
		Contact.Apply();
	}

	/* Below we have a simple implementation of the available "Callback" functions */
function OnZoiperCallFail(call)
	{
		Status(call.Phone + " failed");
	}

	function OnZoiperCallRing(call)
	{
		Status(call.Phone + " ring");
	}

	function OnZoiperCallHang(call)
	{
		Status(call.Phone + " hang");
}

	function OnZoiperCallHold(call)
	{
		Status(call.Phone + " hold");
	}

	function OnZoiperCallUnhold(call)
	{
		Status(call.Phone + " unhold");
	}

	function OnZoiperCallAccept(call)
	{
		Status(call.Phone + " accept");
	}
```

```
function OnZoiperCallReject(call)
{
	Status(call.Phone + " reject");
```

```
    }

    function OnZoiperCallIncoming(call)
    {
        Status(call.Phone + " incoming");
    }

    function OnZoiperAccountRegister(account)
    {
        Status(account.Name + " is registered");
    }

    function OnZoiperAccountUnregister(account)
    {
        Status(account.Name + " is unregistered");
    }

    function OnZoiperAccountRegisterFail(account)
    {
        Status(account.Name + " failed to register");
    }

    function OnZoiperContactStatus(contact,status)
    {
    Status(contact.Name + " is " + status);
    }

    function OnZoiperCallSipHeader(call,sip)
    {
        Status("SIP header counts: " + sip.Count);

        /* This fragment of code demonstrates how to display a specific SIP header (in this
case the "allow" header) */
        var property = sip.Entry("allow");
if (null != property)
        {
            for (var j=0; j<property.Count; j++)
            {
                Status(" Allow: " + property.Value(j));
}
        }

        /* This fragment of code demonstrates how to display all SIP headers */
for (var i=0; i<sip.Count; i++)
        {
            var property = sip.Entry(i);
            Status("Header label: " + property.Name);
for (var j=0; j<property.Count; j++)
            {
                Status("    value: " + property.Value(j));
}
        }
    }
</script>

<!-- Here we define the HTML elements we are going to need in this example -->
<table border=1 width=100%>
    <tr>
        <td rowspan=10>
            <TEXTAREA id="thelog" rows="20" cols="40">
```

```
            </TEXTAREA>
        </td>
    </tr>
    <tr>
```

```
            <td width=100%>
                  Number <input id="number" type="text"/>
                  <button onclick="Dial()">Dial</button>
                  <button onclick="Hold()">Hold</button>
                  <button onclick="Hang()">Hang</button>
</td>
      </tr>
      <tr>
            <td width=100%>
                  DTMF sequence <input id="dtmfsequence" type="text"/>
                  <button onclick="SendDTMFSequence()">Send</button>
            </td>
      </tr>
      <tr>
            <td width=100%>
                  <button onclick="ShowAudioWizard()">Show Audio Wizard</button>
<button onclick="ShowLog()">Show Log</button>                  </td>
      </tr>
      <tr>
            <td>
                  <button onclick="MuteMicToggle()">Mute Mic</button>
                  <button onclick="MuteSpeakerToggle()">Mute Speaker</button>
</td>
      </tr>
      <tr><td> </td></tr>
      <tr><td> </td></tr>
      <tr><td> </td></tr>
      <tr><td> </td></tr>
      <tr><td> </td></tr>
</table>

<!-- Here we define the <object> tag used by Internet Explorer to include Zoiper Web in the
web page. Note that the version info in the "codebase" attribute is very important when a
more recent Zoiper Web is available and should be upgraded -->
<object id="ZoiperA" classid="clsid:BCCA9B64-41B3-4A20-8D8B-E69FE61F1F8B" CODE-
BASE="http://www.zoiper.com/webphone/InstallerWeb.cab#Version=2,7,0,13449" align="left"
width="434" height="236" >

<!-- Here we define the <embed> tag used by the NPAPI based browsers (FireFox, etc)
to include      Zoiper          Web in the web page.-->
<embed id="ZoiperN" type="application/x-zoiper-plugin" align="left" width="434"
height="236"/>
</object>
</body>
</html>
```

Contact us:

e-mail: support@zoiper.com

web: www.zoiper.com

ZOIPER WEB API