PCBest Network AstOCX API Reference

(V1.002)

Copyright 2007-2008 PCBest Network
support@pcbest.net

Index:

1 Introduction

PCBest Network provides AstOCX for Asterisk developers who want to build dynamic FastAGI dialplan on Windows platform. For more information about Asterisk FastAGI, please refer this link: http://www.voip-info.org/wiki-Asterisk+FastAGI.

Our contact information for support:
Email: support@pcbest.net

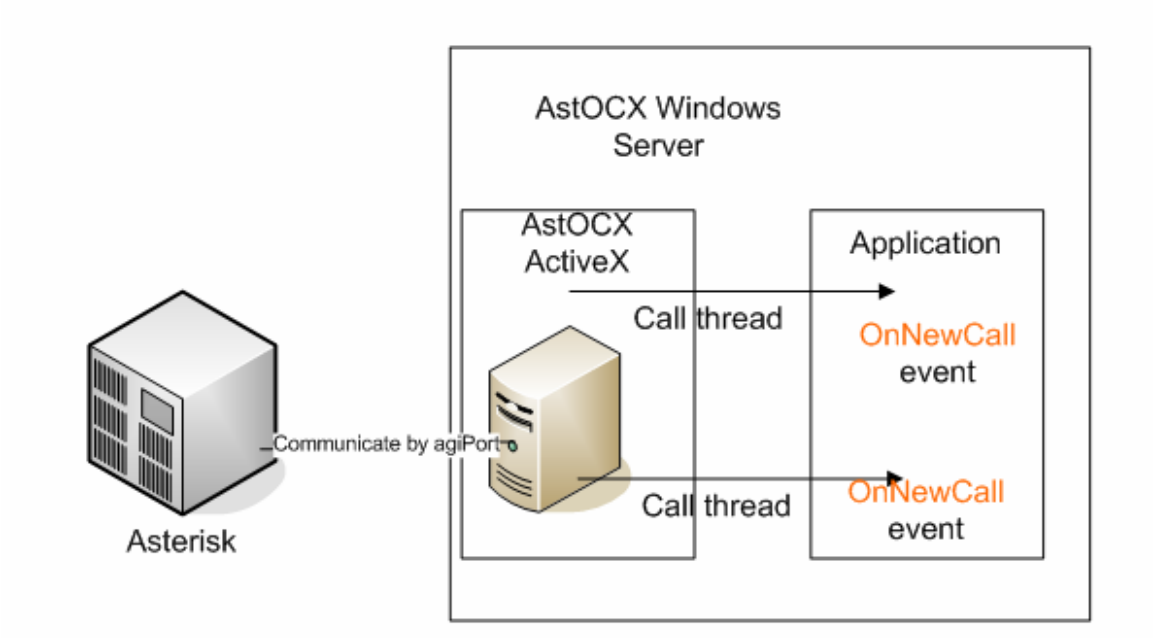
1.1 Key Features and Specifications

AstOCX implements the following AGI commands:

- **answer**: Asserts answer
- **channel status**: Returns status of the connected channel
- **control stream file**: Send the given file, allowing playback to be controled by the given digits, if any. (Asterisk 1.2)
- **database del**: Removes database key/value
- **database deltree**: Removes database keytree/value
- **database get**: Gets database value
- **database put**: Adds/updates database value
- **exec**: Executes a given Application. (Applications are the functions you use to create a dial plan in extensions.conf).
- **get data**: Gets data on a channel
- **get full variable**: Gets a channel variable, but understands complex variable names and builtin variables. (Asterisk 1.2)
- **get option**: Behaves similar to STREAM FILE but used with a timeout option. (Asterisk 1.2)
- **get variable**: Gets a channel variable
- **hangup**: Hangup the current channel
- **noop**: Does nothing
- **receive char**: Receives one character from channels supporting it
- **receive text**: Receives text from channels supporting it
- **record file**: Records to a given file
- **say alpha**: Says a given character string (Asterisk 1.2)
- **say date**: Say a date (Asterisk 1.2)
- **say datetime**: Say a formatted date and time (Asterisk 1.2)
- **say digits**: Says a given digit string
- **say number**: Says a given number
- **say phonetic**: Say the given character string.
- **say time**: Say a time
- **send image**: Sends images to channels supporting it

- **send text**: Sends text to channels supporting it
- **set autohangup**: Autohangup channel in some time
- **set callerid**: Sets callerid for the current channel
- **set context**: Sets channel context
- **set extension**: Changes channel extension
- **set music**: Enable/Disable Music on hold generator, example "SET MUSIC ON default"
- **set priority**: Prioritizes the channel
- **set variable**: Sets a channel variable
- **stream file**: Sends audio file on channel
- **tdd mode**: Activates TDD mode on channels supporting it, to enable communication with TDDs.
- **verbose**: Logs a message to the asterisk verbose log
- **wait for digit**: Waits for a digit to be pressed

1.2 AstOCX Component Architecture



1.3 Samples in AstOCX.zip

There are three samples in the zip now.

AstOcxMFCTest
A C++ MFC sample for using AstOCX.

CSharpAstOCXSample
A C# sample.

VB6Sample
A VB6 sample.

1.4 How to setup development environment

1. Register ocx by command: regsvr32 AstAGI.ocx
2. Run one of sample code, for example, AstOcxMFCTest.exe.
3. Change the dialplan for Asterisk.

In /etc/asterisk/sip.conf, pelase change 'Context' value in general.
For example, set Context = to-ocx-app

In /etc/asterisk/extensions.conf, please add a section for 'to-ocx-app'.
[to-ocx-app]\nexten => _.,1,AGI(agi://your.windows.ip.address:8916/service1)

4. Choose your dial plan on the screen.
5. Use X-Lite or PCBest Network's web SIP phone to call your Asterisk box.
6. Enjoy the ActiveX control, and contact us by http://www.pcbest.net/contactus.htm

2 Programming Guide

2.1 Initialization code

No matter what language you are using, we suggest you to use the following SDK initialization code for your programming convenience.

```
void InitAstOCX()
{
    m_AstAGI1.SetAgiPort(8916);
    m_AstAGI1.SetLogFileName("c:\\agi-ocx.txt");
    m_AstAGI1.InitSrv();
}
```

```
void FreeAstOCX()
{
    m_AstAGI1.FreeSrv();
}
```

2.2 Form Load and Unload

```
void Form_Load()
{
        InitAstOCX();
}

void Form_Unload()
{
        FreeAstOCX();
}
```

2.3 Working with AstOCX functions

The AstOCX provides a lot of functions for application level. You can find one function for each AGI command. Most of them have same name. There is only one event for AstOCX, which is OnNewCall. Every time when there is a new call coming in, this event is triggered. Basically your code will be like:

```
OnNewCall()
{
        //Get AGI parameters by using GetAGIVars
        m_AstAGI1.GetAGIVars(CallUniqueID, "agi_channel");
        m_AstAGI1.GetAGIVars(CallUniqueID, "agi_callerid");
        m_AstAGI1.GetAGIVars(CallUniqueID, "agi_context");
        m_AstAGI1.GetAGIVars(CallUniqueID, "agi_extension");
        ……

        //Send Command 1 to do thing 1
       long ret = m_AstAGI1.Command1;

        //check the result of command1 by ret or one of the following functions
```

m_AstAGI1. GetCmdResCode, GetCmdResText, GetCmdResEndpos, GetCmdResData, or GetCmdResRaw

```
//if it is ok, do command 2
if(ret == 200)
m_AstAGI1.Command2

//check command2 result
……

……

//End. Hungup the call
m_AstAGI1.Hungup
```

}

3 AstOCX API Reference

3.1 InitSrv

Initialize AstOCX AGI server. It will open the a TCP port for listening. Before you use this function, you should have already set agiPort.
Sample code:
AstOCX.agiPort = 8916;
AstOCX.InitSrv();

3.2 FreeSrv

Free AstOCX. This will make it close the TCP port on the machine.

3.3 GetCmdResCode, GetCmdResText, GetCmdResData, GetCmdResEndpos and GetCmdResRaw

Those functions present the result from a command. Everytime, after you send a command to Asterisk, and the executing of command is done, you will receive command result. Some results are simple, just like: 200 result=0 or 200 result=-1
Sample:
AstOCX.Answer(..)
If(AstOCX.GetCmdResCode() = 200) …..

3.3 Answer

Answer a call.
AGI command: ANSWER
Format: long Answer(long CallUniqueID)
CallUniqueID: from OnNewCall
Return: result code

3.3 AutoHungup

AGI command: SET AUTOHANGUP
Format: long AutoHungup(long CallUniqueID, long TimeValue)
CallUniqueID: from OnNewCall
TimeValue: in seconds.
Return: result code

3.4 ChannelStatus

AGI Command: CHANNEL STATUS [<channelname>]
Format: long ChannelStatus(long CallUniqueID, LPCTSTR Channel)
CallUniqueID: from OnNewCall
Channel: channel name
Return: result code

3.5 Exec

AGI Command: EXEC <application> <options>

Format: long Exec(long CallUniqueID, LPCTSTR AppName, LPCTSTR Options)
CallUniqueID: from OnNewCall
AppName: Name of application
Options: Parameters for application
Return: result code

3.6 GetVariable

AGI Command: GET VARIABLE <variablename>
Format: long GetVariable(long CallUniqueID, LPCTSTR VarName)
CallUniqueID: from OnNewCall
VarName: Name of variable
Return: result code

3.7 Hungup

AGI Command: HANGUP [<channelname>]
Format: long Hungup(long CallUniqueID, LPCTSTR Channel)
CallUniqueID: from OnNewCall
Channel: Channel ID
Return: result code.

3.8 ReceiveChar

AGI Command: RECEIVE CHAR <timeout>
Format: long ReceiveChar(long CallUniqueID, long Timeout)
CallUniqueID: from OnNewCall
Timeout: in seconds
Return: result code.

3.9 RecordFile

AGI Command: RECORD FILE <filename> <format> <escape digits> <timeout> [offset samples] [BEEP] [s=<silence>]
Format: long RecordFile(long CallUniqueID, LPCTSTR FileName, LPCTSTR Format, LPCTSTR EscapeDigits, long Timeout, long Beep, long Silence, long Offset)
CallUniqueID: from OnNewCall
FileName: file name
Format: file format
EscapeDigits: digits to stop recording
Timeout: in seconds
Beep: if beep
Offset: byte offset
Return: result code

3.10 SayDigits

AGI Command: SAY DIGITS <number> <escape digits>
Format: long SayDigits(long CallUniqueID, LPCTSTR Digits, LPCTSTR EscapeDigits)
CallUniqueID: from OnNewCall
Digits: digits to say
EscapeDigits: digits to escape
Return: result code

3.11 SayPhonetic

AGI Command: SAY PHONETIC <string> <escape digits>
Format: long SayPhonetic(long CallUniqueID, LPCTSTR TextStr, LPCTSTR EscapeDigits)
CallUniqueID: from OnNewCall
TextStr: text to say
EscapeDigits: digits to escape
Return: result code

3.12 SayNumber

AGI Command: SAY NUMBER <number> <escape digits>
Format: long SayNumber(long CallUniqueID, long Number, LPCTSTR EscapeDigits)
CallUniqueID: from OnNewCall
Number: number to say
EscapeDigits: digits to escape
Return: result code

3.13 SayTime

AGI Command: SAY TIME <time> <escape digits>
Format: long SayTime(long CallUniqueID, long UnixTimeVal, LPCTSTR EscapeDigits)
CallUniqueID: from OnNewCall
UnixTimeVal: time to say
EscapeDigits: digits to escape
Return: result code

3.14 SendImage

AGI Command: SEND IMAGE <image>
Format: long CAstAGICtrl::SendImage(long CallUniqueID, LPCTSTR ImageFile)
CallUniqueID: from OnNewCall
ImageFile: Image file name
Return: result code

3.15 SendText

AGI Command: SEND TEXT "<text to send>"
Format: long SendText(long CallUniqueID, LPCTSTR TextStr)
CallUniqueID: from OnNewCall
TextStr: text to send
Return: result code

3.16 SetCallerID

AGI Command: SET CALLERID <number>
Format: long SetCallerID(long CallUniqueID, LPCTSTR CallID)
CallUniqueID: from OnNewCall
CallID: New caller id
Return: result code

3.17 SetContext

AGI Command: SET CONTEXT <desired context>
Format: long SetContext(long CallUniqueID, LPCTSTR Context)
CallUniqueID: from OnNewCall
Context: New context to set
Return: result code

3.18 SetExtension

AGI Command: SET EXTENSION <new extension>
Format: long SetExtension(long CallUniqueID, LPCTSTR Extension)
CallUniqueID: from OnNewCall
Extension: New extension
Return: result code

3.19 SetPriority

AGI Command: SET PRIORITY <num>
Format: long SetPriority(long CallUniqueID, long PriorityVal)
CallUniqueID: from OnNewCall
PriorityVal: value
Return: result code

3.20 SetVariable

AGI Command: SET VARIABLE <variablename> <value>
Format: long CAstAGICtrl::SetVariable(long CallUniqueID, LPCTSTR VarName, LPCTSTR VarValue)
CallUniqueID: from OnNewCall
VarName: name of variable
VarValue: value of variable
Return: result code

3.21 StreamFile

AGI Command: STREAM FILE <filename> <escape digits> [sample offset]
Format: long StreamFile(long CallUniqueID, LPCTSTR FileName, LPCTSTR EscapeDigits, long Offset)
CallUniqueID: from OnNewCall
FileName: name of file to play
EscapeDigits: digits to escape
Offset: offset byte to start playing
Return: result code

3.22 TddMode

AGI Command: TDD MODE <on|off|mate>
Format: long TddMode(long CallUniqueID, LPCTSTR Setting)
CallUniqueID: from OnNewCall
Setting: on, off, or mate
Return: result code

3.23 Verbose

AGI Command: Verbose(<message> [level])
Format: long CAstAGICtrl::Verbose(long CallUniqueID, LPCTSTR Message, long Level)
CallUniqueID: from OnNewCall
Message: message to log
Level: message verbose level
Return: result code

3.24 WaitForDigit

AGI Command: WAIT FOR DIGIT <timeout>
Format: long WaitForDigit(long CallUniqueID, long Timeout)
CallUniqueID: from OnNewCall
Timeout: timeout in seconds
Return: result code

3.25 DBPut, DBGet, DBDel, DBDelTree

AGI Command:
DATABASE PUT <family> <key> <value>
DATABASE GET <family> <key>
DATABASE DEL <family> <key>
DATABASE DELTREE <family> [keytree]

3.26 Noop

AGI Command: NOOP
Format: long Noop(long CallUniqueID)
CallUniqueID: from OnNewCall
Return: result code

3.27 SetMusic

AGI Command: SET MUSIC <on|off> <class>
Format: long SetMusic(long CallUniqueID, long Enabled, LPCTSTR MusicClass)
CallUniqueID: from OnNewCall
Enabled: 1 or 0
MusicClass: music class
Return: result code

3.28 ExecAbsoluteTimeout, ExecSetLanguage, ExecDial, ExecGoto

Convenient functions for exec. Please refer to dialplan’s applications for those functions.

3.29 GetAGIVars

Get variables passed to AGI scripts.
Here are some of the variables passed to AGI scripts:

agi_channel — calling channel
agi_callerid — CALLERID(num)
agi_calleridname — CALLERID(name)
agi_context — dialplan context from which script was called
agi_extension — dialplan extension from which script was called
agi_priority — dialplan priority from which script was called
agi_language — CHANNEL(language)

Global variables are not passed to the AGI script in this manner. You must get them using the "get variable" AGI command.

3.30 WriteLog

Write a text into AstOCX log file.
Format: void WriteLog(long LogLevel, LPCTSTR LogInfo)
LogLevel: log level. 0 fatal, 1 error, 2 waring, 3 trace, 4 info
LogInfo: text to log

3.31 TSleep

Sleep current thread for a while.
Format: long TSleep(long CallUniqueID, long MilliSeconds)
CallUniqueID: from OnNewCall
MilliSeconds: milliseconds to sleep(do nothing)