Technical Note

Com.X CLI and AMI Interoperability guide

Includes extension and pin blocking instructions

Version 1.0, 19 November 2010

Document History

Version	Date	Description of Changes
1	10/11/19	Document created.

Table of Contents

1 Introduction

1.1 Overview

This document is intended for the Com.X PBX and Gateway integrators and contains details of the APIs available for command, and event interop with Com.X systems. Instructions for blocking extensions and pins in pin sets are also included.

Integrators in environments such as hotels, guest houses, gated communities, enterprises and others may require the ability to dynamically block or unblock extensions or pins from dialing outbound routes. Such a requirement may be based on a set of criteria, such as booking in or signing out from a hotel, exceeding a budgetary limit on an extension or pin, etc.

TMS and hospitality software that monitors and manages call records, account balances and reservation status may need to interact with the Com.X system in order to facilitate and enact blocking and unblocking of extensions and pins.

This document details the API meeting these requirements.

2 Issuing commands to the Com.X system

2.1 CLI via login shell

The Com.X system provides a login shell via the SSH protocol. Users can log into the system using an SSH-enabled command shell or tool (such as Putty).

Once logged in, commands can be issued to the Com.X system using the `asterisk` command with the remote login and execute parameters specified. The command requires root access. To see the number of channels in progress, for example, one would issue the command below from the shell:

```
sudo asterisk -rx "core show channels"
```

2.2 sudo

When using `sudo`, the user might be prompted to enter a password to enable super-user privileges. The password would be the same as the user's normal password. Only users allowed to become super-users are allowed the use of the `sudo` command.

To add a user to the sudo group, do:

```
sudo addgroup username sudo
```

To enable a user to execute specific (or any) commands using `sudo` without being prompted for a password, add the user to the `/etc/sudoers file`

To limit a user to specific set of commands, add the commands to the sudo list in `/etc/sudoers`

To safely edit `/etc/sudoers`, use the `sudo visudo` command. `visudo` checks for syntax problems, avoiding catastrophic problems. If the `/etc/sudoers` file becomes corrupt, you will have to boot off an installation CD and open a root console to manually correct it before being able to perform any actions as super-user.

The order of entries in `/etc/sudoers` is important, with later entries for the same user replacing earlier entries.

Examples:

```
%admin ALL=(ALL) ALL

%comma ALL=(root) NOPASSWD: /usr/sbin/asterisk, /etc/init.d/commamgr
```

2.3 AMI

2.3.1 Overview

The Asterisk Manager Interface allows a user or program to interactively monitor PBX events and issue commands over a simple TCP/IP connection.

The AMI interface utilizes a key : value line-based protocol, with commands indicated by two consecutive carriage returns.

Connections to the Com.X for AMI access are made on TCP port 5038 by default. To see which AMI clients are connected, issue the `manager show connected` command from the CLI.

AMI clients communicate with the server using *Action* messages, and the server responds with *Response* or *Event* messages.

2.3.2 Configuring access

Once connected to AMU, in order to issue commands and receive events, a user must log into the system. User login configurations are stored in the `/etc/asterisk/manager.d/` directory on the Com.X and take the form of configuration files named <user>.conf The configuration file details the username, allowed IP addresses and permissions for the user. For example:

```
[amiuser]
secret = amitest
deny = 0.0.0.0/0.0.0.0
permit = <ip-of-machine>/255.255.255.0
read = system,call,log,verbose,command,agent,user
write = system,call,log,verbose,command,agent,user
```

2.3.3 Logging in

To log in, issue the following command set, enacted by two carriage returns:

```
Action: Login

ActionID: 1

Username: amiuser

Secret: amitest
```

Success is indicated by:

```
Response: Success

ActionID: 1

Message: Authentication accepted
```

2.3.4 Issuing commands

Commands issued via AMI can be assigned *ActionID*s, which are included in responses received from the server. Since the order in which responses are received are not guaranteed to be sequential (some commands may take longer to complete than others) the *ActionID* is useful to correlate a response with the command that prompted it.

The list of commands available via the AMI interface can be retrieved using the `manager show commands` CLI command. Details on an individual command are available using `manager show command <command>`.

For example:

```
FSN1X00004*CLI> manager show command MailboxCount
Action: MailboxCount
Synopsis: Check Mailbox Message Count
Privilege: call,all
Description: Checks a voicemail account for new messages.
Variables: (Names marked with * are required)
        *Mailbox: Full mailbox ID <mailbox>@<vm-context>
        ActionID: Optional ActionID for message matching.
Returns number of new and old messages.
        Message: Mailbox Message Count
        Mailbox: <mailboxid>
        NewMessages: <count>
        OldMessages: <count>
```

To issue a command, specify the Action, ActionID and any required parameters. Interpret the response as required. For example:

```
Action: MailboxCount
ActionID: 10092
Mailbox: 2000

Response: Success
ActionID: 10092
Message: Mailbox Message Count
Mailbox: 2000
NewMessages: 1
OldMessages: 0
```

To issue a CLI command using AMI, use:

```
Action: Command
ActionID: <id>
Command: <CLI command>
```

3 Blocking Extensions

3.1 Enabling the feature

In order to block extensions, create (or merge into) /etc/asterisk/extensions_custom.conf:

```
[macro-dialout-trunk-predial-hook]
exten => s,1,NoOp("Status ${DB(AMPUSER/${CALLERID(number)}/BLOCKED)} for $
{CALLERID(number)}")
exten => s,n,GotoIf($["${DB(AMPUSER/${CALLERID(number)}/BLOCKED)}" = "YES"]?disallow)
exten => s,n,Set(pre_num=${CUT(OUT_${DIAL_TRUNK},$,1)})
exten => s,n,GotoIf($["${pre_num:0:9}" != "AMP:mISDN"]?notmisdn)
exten => s,n,Set(misdn=${CUT(pre_num,/,2)})
exten => s,n,misdn_check_l2l1(${misdn},5)
exten => s,n(notmisdn),NoOp()
exten => s,n,MacroExit
exten => s,n(disallow),Set(PREDIAL_HOOK_RET="BYPASS")
```

After uploading the extension, reload the dial plan by issuing the dialplan reload CLI command.

3.2 Behaviour

This dial-plan hood executes before a call is placed over a trunk (i.e. internal calls excluded) and verifies whether the extension in question has been blocked. Verification is by means of a database lookup. If the extension is blocked, the call will be disallowed. An “All circuits are busy now” message will be played and the call terminated.

Note that this modification is merged into an existing custom dialplan entry that improved Basic Rate ISDN performance.

To use this customization, issue the following commands using the CLI or AMI:

<exten> indicates the calling extension

3.3 Commands

To block an extension:

```
database put AMPUSER/<exten> BLOCKED YES
```

To unblock an extensionn:

```
database del AMPUSER/<exten> BLOCKED
```

4 Blocking pins

4.1 Enabling the feature

To enable this customization, place the following dial-plan code in `/etc/asterisk/extensions.conf` before:

```
#include extensions_override_freepbx.conf
```

Dialplan modification:

```
[macro-pinsets]
exten => s,1,NoOp(Custom auth initiated with User: ${AMPUSER} Pinset index: ${ARG1})
; Check whether the user is exempt from pins
exten => s,n,GotoIf($["${DB(AMPUSER/${AMPUSER}/pinless)}" != "NOPASSWD"]?doauth)
exten => s,n,MacroExit()
; Perform auth
exten => s,n(doauth),Playback(agent-pass&beep)
exten => s,n,Read(USERPIN)
exten => s,n,System(/usr/share/asterisk/bin/auth.sh ${USERPIN} /etc/asterisk/pinset_$
{ARG1})
exten => s,n,GotoIf($["${SYSTEMSTATUS}" == "APPERROR"]?failauth)
exten => s,n,GotoIf($["${DB(FSN/PINS/${USERPIN}/BLOCKED)}" = "YES"]?blockedpin)
exten => s,n,Playback(auth-thankyou)
exten => s,n,ResetCDR()
exten => s,n,GotoIf($[${ARG2}==0]?exitpoint)
exten => s,n,Set(CDR(accountcode)=${USERPIN})
exten => s,n(exitpoint),MacroExit()
exten => s,n(failauth),Playback(astcc-account-number-invalid)
exten => s,n,Wait(2)
exten => s,n,Hangup()
exten => s,n(blockedpin),Playback(im-sorry&your-account&is&ha/locked)
exten => s,n,Wait(2)
exten => s,n,Hangup()
; end of [macro-pinsets]
```

Create `/usr/share/asterisk/bin/auth.sh` with the following permissions: `-rwxr-xr-x`

```
#!/bin/bash

# Check for valid input
if [ "" == "$2" ]; then
    exit 1
fi

LEADING=$(echo $2 | cut -c1-1)
#Is this an auth list?
if [ "$LEADING" == "/" ]; then
  RESULT=$(grep $1 $2)
  if [ "$1" != "$RESULT" ]; then
    exit 1
  else
    exit 0
  fi
fi

# Verify against the expected value
if [ "$1" == "$2" ]; then
  exit 0;
else
  exit 1;
fi
```

4.2 Behaviour

When an outbound call is placed over a trunk with a pin set associated, the extension is checked to verify whether pin bypass is enabled for the extension. If so, the call is allowed. If pin bypass is not enabled, the user is prompted for a pin. The pin is then checked against the pin set, and if it matches, a check is made against the database to verify that the pin is not blocked. If the pin is blocked, the call is disallowed with a message indicating that the pin is locked. If the pin is not blocked and is in the pin set, the call is allowed. If the pin is not in the pin set, the call is not allowed.

This feature only affects outbound routes with pin sets assigned. Outbound routes with a password assigned follows the default behaviour.

The user is prompted to enter their pin only once (no retries).

The user must enter the pin after the beep

CDR recording of the pin used is dependent on the "Record in CDR" flag on the pin set.

If pins overlap (e.g. 1234 and 12345), only the first pin in the list will always be available, so best avoid this by keeping all pins the same length

4.3 Commands

To use this customization, issue the following commands using the CLI or AMI:

<pin> indicates the user pin

<exten> indicates the calling extension

To block a pin:

```
database put FSN/PINS/<pin> BLOCKED YES
```

To unblock a pin:

```
database del FSN/PINS/<pin> BLOCKED
```

To enable pin bypass for an extension:

```
database put AMPUSER/<exten> pinless NOPASSWD
```

To disable pin bypass for an extension:

```
database del AMPUSER/<exten> pinless
```