AMI application interface

AMI — powerful and convenient application interface (API) of Asterisk to control the system from external applications.
Due to AMI external applications can connect to Asterisk using TCP protocol, initiate command execution, fetch execution result, and get notifications about various events in real-time mode. You can use these mechanisms for example in following cases:

- You need to get system's state
- The amount of active subscribers
- Initiate CLI commands remotely
- Initiate calls
- etc

AMI is often used to integrate business-processes and systems, CRM software (Customer Relationship Managment). It also could be used for various applications, such as automatic dialers and click-to-call systems.

Asterisk control often is done from CLI console.
However when using AMI you don't need direct access to the server, on which Asterisk is launched.
AMI — is the simplest tool, which can become a very powerful and flexible mean of integration with other software products in developer's hands. It gaves possibility to developers to use information generated by Asterisk in real-time mode.

How AMI works

Between Smartswitch server and client application a simple line-by-line protocol is used, each line of which contains of 2 strings:

- key - a word which describes the type of information which is contained in current line. A key word is not unique and could be seen several times in the frame of one packet
- value - the parameter value

The key word is delimited from value by colon.
Below we will use a term «packet» to refer to a «key:value» construction set, which is delimited by CRLF and ended with additional CRLF sequence.

The protocol has the following characteristics:

- Before issuing commands to Asterisk, you must establish a manager session (see below).
- Packets may be transmitted in either direction at any time after authentication.
- The first line of a packet will have a key of "Action" when sent from the client to Asterisk, but "Event" or "Response" when sent from Asterisk to the client.
- The order of lines within a packet is insignificant, so you may use your favorite programming language's native unordered dictionary type to efficiently store a single packet.
- CRLF is used to delimit each line and a blank line (two CRLF in a row) indicates the end of the command which Asterisk is now expected to process.

AMI accepts connections on a network port (TCP 5038 by default). Client applications connects to AMI through this port and authenticates, after this Asterisk will respond to requests, and forward notifications about subsystem changes.

Packet Types

The type of a packet is determined by the existence of one of the following keys:

- **Action**
  A packet sent by the connected client to Asterisk, requesting a particular Action be performed.
  There are a finite (but extendable) set of actions available to the client, determined by the modules presently loaded in the Asterisk engine.
  Only one action may be outstanding at a time.
  The Action packet contains the name of the operation to be performed as well as all required parameters.

- **Response**
  The response sent by Asterisk to the last action sent by the client.

- **Event**
  Data pertaining to an event generated from within the Asterisk core or an extension module.

Generally the client sends Action packets to the Asterisk server, the Asterisk server performs the requested operation and returns the result (often only success or failure) in a Response packet. As there is no guarantee regarding the order of Response packets the client usually includes an ActionID parameter in every Action packet that is sent back by Asterisk in the corresponding Response packet. That way the client can easily match Action and Response packets while sending Actions at any desired rate without having to wait for outstanding Response packets before sending the next action.

Event packets are used in two different contexts: On the one hand they inform clients about state changes in Asterisk (like new channels being created and hung up or agents being logged in and out) on the other hand they are used to transport the response payload for actions that return a list of data (event generating actions). When a client sends an event generating action Asterisk sends a Response packed indicating success and containing a "Response: Follows" line. Then it sends zero or more events that contain the actual payload and finally an action complete event indicating that all data has been sent. The events sent in response to an event generating action and the action complete event contain the ActionID of the Action packet that triggered them, so you can easily match them the same way as Response packets. An example of an event generating action is the Status action that triggers Status events for each active channel. When all Status events have been sent a terminating a StatusComplete event is sent.

The next CLI command (Tab auto-completion works) can help to get full AMI command list, which are available at your Asterisk version:

*CLI> manager show commands

Response packets. As was written above, serve as answers to sent commands.
Only one response is sent per command and it can contain several meanings:

- "Success" - the action was successful and all information in contained in this packet
- "Error" - an error happened, the full description is inside "Message" header
- "Follows" - the result of execution will be send in next Event packets

Event packets (notifications) are applied in two contexts. On one hand they inform client about Asterisk subsystem status change, on the other hand - they carry a data set, which is returned by Asterisk in response to some Action.

When client sends Action packet, Asterisk can (in cases when there is a need to return several records of the same type) send Response packet, which contains only record «Response: Follows». Then Asterisk sends some amount of events which contain data and, finally, an event which indicates that all data has been transmitted. All generated Event packets contain ActionID of the initial Action packet, which has initiated the query. Therefore, you can easily handle it, just as Response packets. An example of event which is generated by Action — is Action Status, which initiates Status event for each of active channels. When all Status events are sent, StatusComplete event is sent.

Events are created by various structural parts of Asterisk (channels SIP/IAX2/..., CDR, dialplan, various parts of kernel). The main feature, which is performed by events, is to allow external attached system to get information from Asterisk, gathering this information, analyzing it and acting accordingly to results.

Connecting to AMI interface

To connect to AMI, you can use *telnet* application.
This is how system greeting looks like:

```
$ telnet 127.0.0.1 5038
Trying 127.0.0.1...
Connected to localhost.

Escape character is '^]'. Asterisk Call Manager/1.1
```

Then enter:

```
Action: login
Username: admin
Secret: passwd1234
```

and Enter 2 time, which is equivalent to CRLF
After this you should see this response (Response packet):

```
Response: Success
Message: Authentication accepted
```

You can see all users, which are connected to AMI interface at the moment, from Asterisk console:

```
*CLI> manager show connected
```

The password change and command execution 'manager reload' doesn't influence to access right of already established AMI sessions.

If you don't need events which are generated by Asterisk (there can be lots of them, which could make reading responses in Asterisk console more difficult), then you can include one more record in parameters which are passed during authentication process:

```
Events: off
```

This parameter will turn off sending of Event packets from the side of server during current connection.

To terminate AMI session, need to type:

```
Action: logoff
```

After this we'll see in telnet session:

```
Response: Goodbye
Message: Thanks for all the fish.
```

During AMI session termination, in Asterisk console we'll see following:

```
asterisk*CLI> == Manager 'admin' logged off from 127.0.0.1
```

Notice.
Of course AMI interface is not designed for administrator to manually logging in using telnet client and executing commands. This example is demonstrated solely to show features and the workflow of exchanged Action, Event and Response packets. However this approach could come in handy when learning and debugging of some particular command.

To see all commands which AMI supports, type from console:

```
asterisk *CLI> help manager
```

Русский перевод