Asterisk Manager and GUI Interfacing in Large Environments

Presented by Matt Florell
Chief Architect of the astGUIclient project

Manager API and GUI Topics

- Why use the manager API?
- Manager API capacity
- Multi-server Environments
- Central Queue vs. Individual Connections
- How a Central Queue Operates
- Why use MySQL for the Central Queue?
- Real-World Example of Queue in Action
- astGUIclient Design and Functionality
- Capacity Limits and Trade-offs
- The Future

Why Use the Manager API?

- Allows for actions that are either hard or impossible to program in a dialplan or AGI
  - Start/stop recording of existing call
  - Place calls from within meetme conferences
  - Originate call to an AGI and an outside number
  - Redirect existing calls to different channels
- Allows for Complex Client Applications
  - Windows and X-based GUI applications
    - Receptionist consoles
    - ACD/inbound CRM applications and popups
    - Outbound/predictive/auto-dialing call center applications
  - Web-based applications
    - Initiating conferences, remote worker telecommuting
    - Real-time usage reports

Manager API Capacity

- Greatly dependent upon Asterisk and server load
- At high loads data flow can “pause” on the manager interface for up to several seconds
- More manager connections means more chance for pauses no matter the load
- Customizing manager.conf for each connection's needs helps reduce unneeded data transfer and reduces strain on system

Multi-Server Environments

- Large installations that have high-capacity phone usage may need multiple Asterisk servers to run optimally, adding complexity to Manager API usage
- A high-end server running Asterisk can handle 50 concurrent SIP to Telco conversations reliably
- Options for server to server connections
  - IAX – Native Asterisk connection
  - PRI crossover – T1 to T1, no transcoding
  - SIP – harder to setup, wide VOIP standard

Central Queue vs. Individual Connections

- Central Queue:
  - Adds delay in execution (< 0.5 sec)
  - Creates a single point of failure
  - Easily handles interfacing with multiple Asterisk servers
  - Easier/more organized way of keeping track of calls
  - Only needs one local connection for output of all activity
  - Can use new connection for each initiation of a new action preventing action backlogs
- Individual Connections:
  - Can load server more
  - Not as fault tolerant
  - Must follow all call progresses for each connection meaning more work for client app
  - More prone to lock up or freeze

How a Central Queue Operates
Server-side Operations

- Elements running on the Asterisk server
  - Channel state updater – This does nothing but ask for the list of live channels on the Asterisk server(Show Channels) a hundred times a minute to keep updated list of live channels
  - Action sender – Constantly checks for new Actions to be sent to Asterisk servers and starts child processes to send them to the Manager Interface
  - Action Listener – Looks for the Actions that were sent with the sender in the Manager output and updates their status in the Queue

How a Central Queue Operates
Client-side Operations

- Client app can grab the list of live channels at any time
- Client can initiate new actions and look for a response from Central Queue without ever interfacing with the Asterisk server
- Client does not need to keep active connection with the Central Queue like it would with the Manager Interface, making development platform options more flexible

How a Central Queue Operates
Call-Flow Example

- Here is a flow of how a call is initiated through a Central Queue
  1) Client connects to Queue and inserts parameters of the call to be placed
  2) Server sender app sees new Queue entry and initiates child process to connect to the Manager and send the new Action
  3) Server listener app sees output from Manager and matches it up to the Queue entry and up-dates that record's status
  4) Client looks at Queue record it sent and sees that the call went through

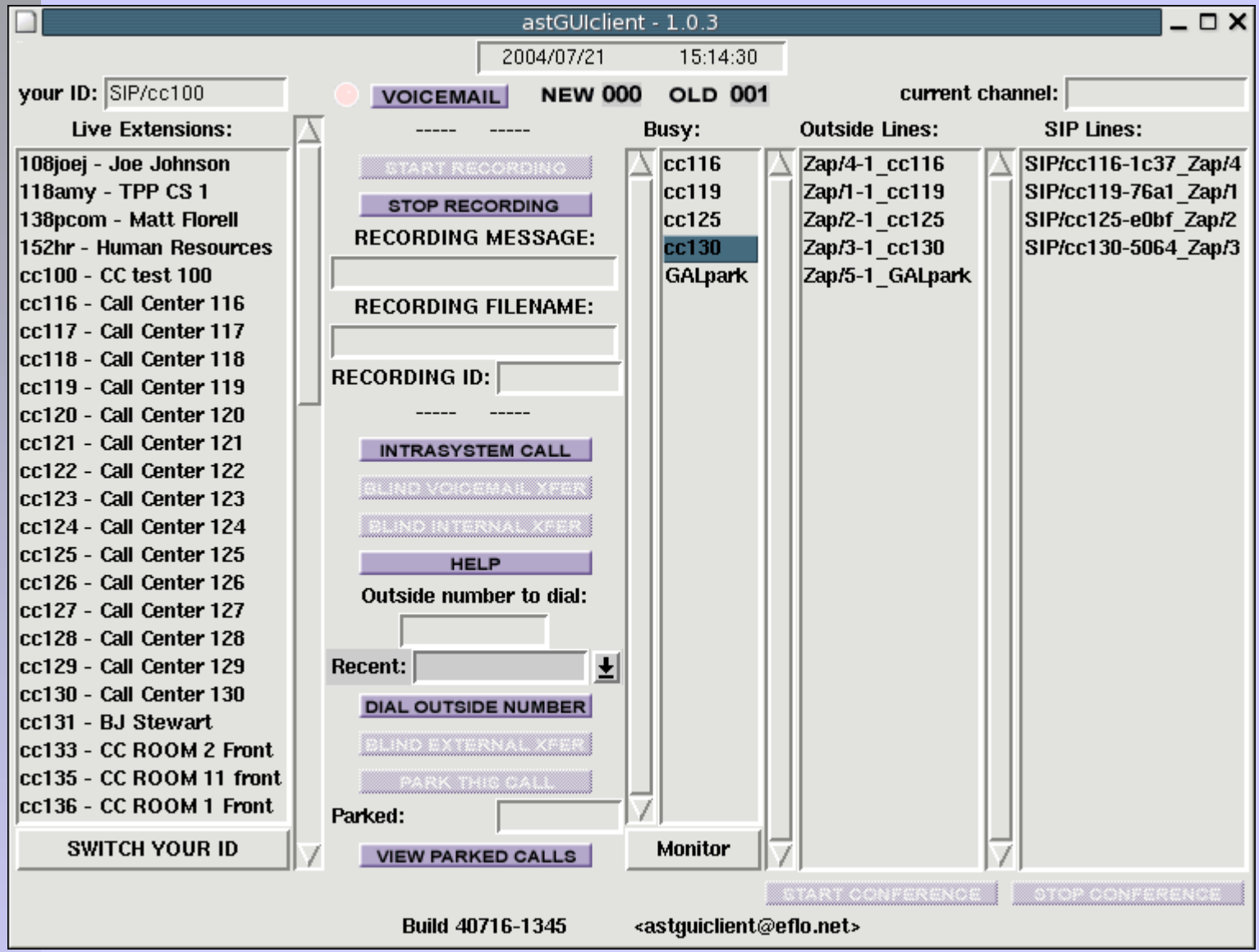
Why Use MySQL for the Central Queue?

- Speed – MySQL is a very fast system for in-formation exchange
- Compatibility – Client libraries are widely available and are very light-weight, and server runs on UNIX and Win32
- Capacity – You can have several hundred concurrent connections on a single MySQL server
- Ease of use – MySQL is simple to Set up, Administrate and Write custom Queries for

Real-World Example of a Central Queue in Action

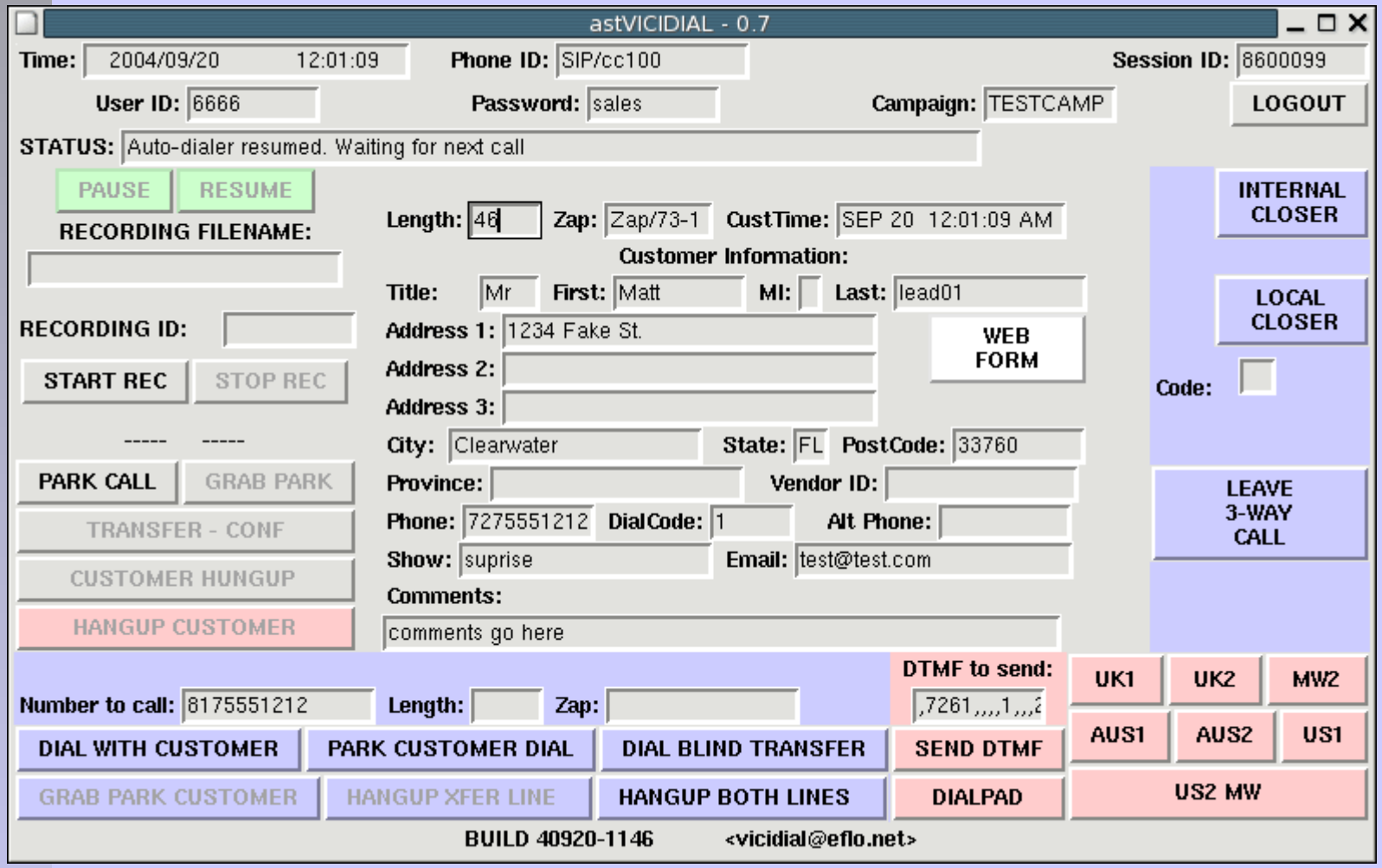
- Corporate and Call Center Environment with over 150 Employees
- Five Asterisk servers with 16 T1s connected
- One MySQL server acting as Central Queue
- 150 SIP telephone devices
- Inbound/Outbound telemarketing with custom GUI client apps for CRM and auto-dialing
- Local and Remote Customer service and Sales Agents using Web interface for call manipulation and information exchange
- Real-time stats and reports on system operation and agent performance
- Receptionist console and corporate console with click-to-record and click-to-conference

astGUIclient functionality



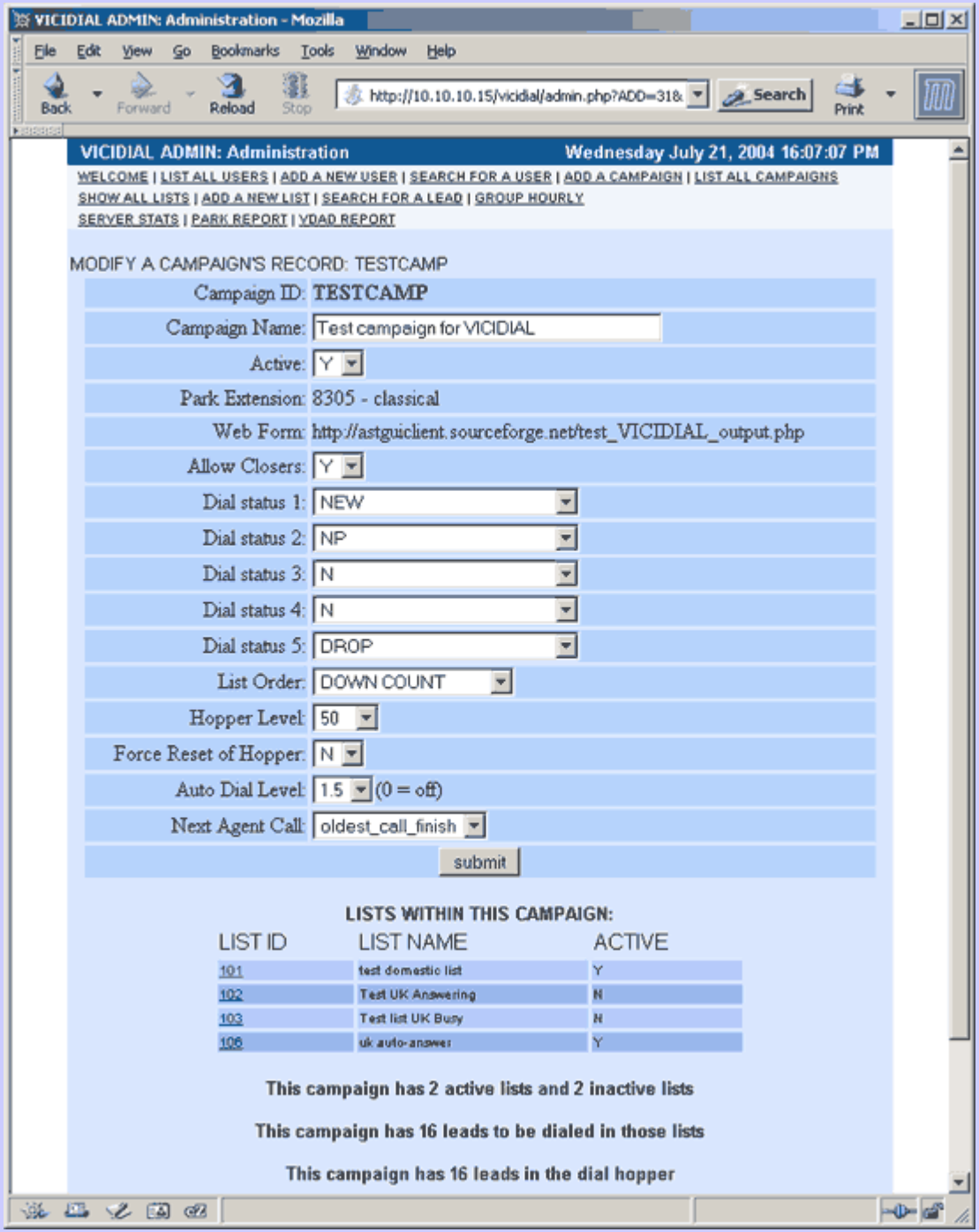
- List live channels
- Recording
- Blind Monitoring
- Voicemail Transfer
- Internal Transfer
- External Transfer
- Call Parking
- Forced Hangup
- Conferencing with up to six external channels
- List of recent calls out
- Click to check voicemail
- Inbound call popup

VICIDIAL screenshot

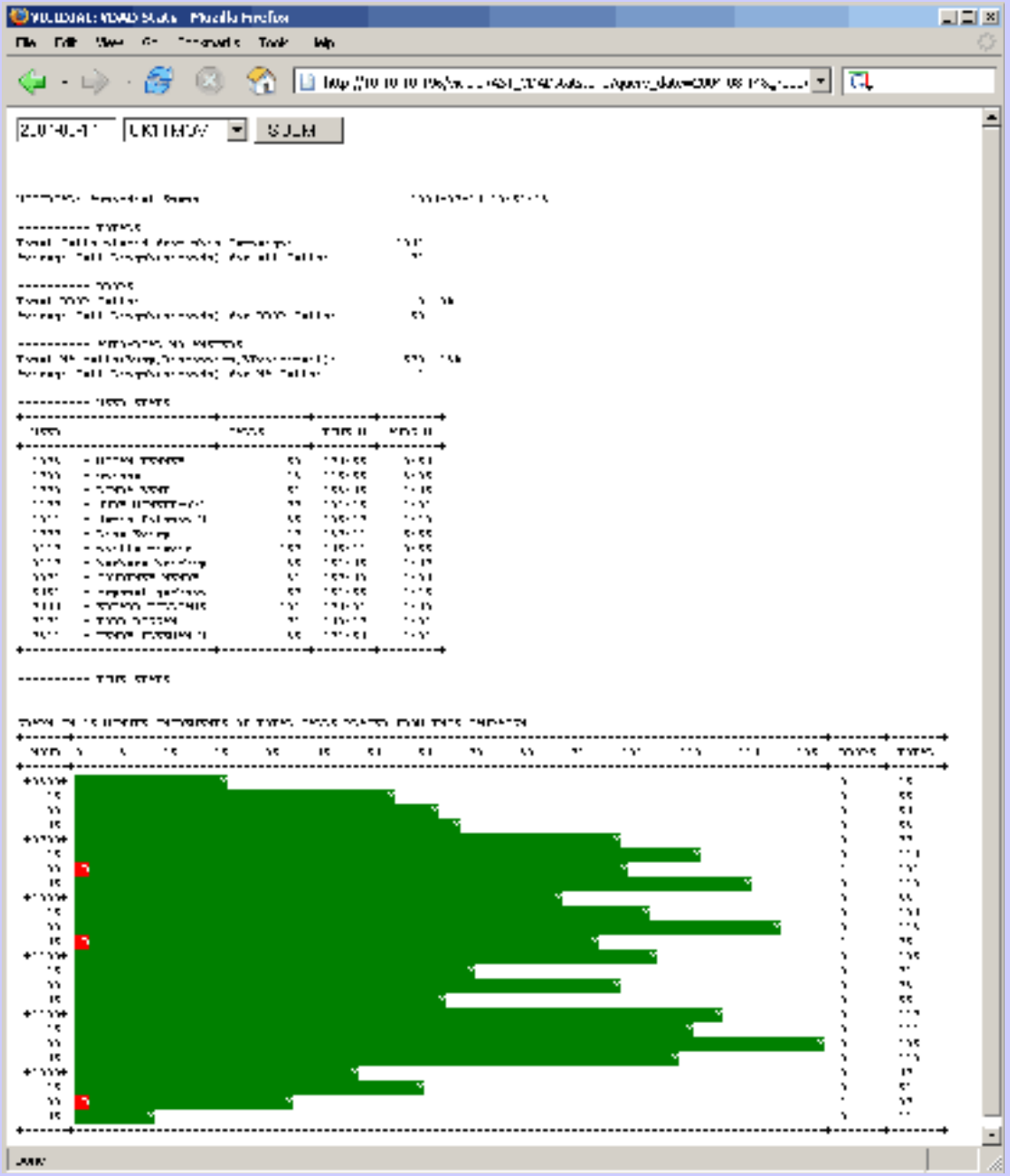


- Dial by list
- Call Recording
- Third party conference, transfer and drop
- DTMF macros
- Call Parking
- Custom on-hold music
- Click-to-dial or predictive dialing
- Web-based closers can be local or remote
- Time zone dialing restriction available

Adminsitration screenshot



Reports screenshot



Capacity Limits and Trade-offs

- Example Central Queue handles:
  - Over 100,000 sent Manager actions daily
  - Over 30,000 calls in and out daily
  - Over 400 concurrent Asterisk channels
  - Five live Asterisk servers
  - 120 connected client applications
- Trade-offs of using Central Queue:
  - Only Manager traceable tag is “CallerID” which is a big negative especially for PRI users with customizable CallerID out
  - Central Queue is a single point of failure
  - Slight delay of action execution, not very notice-able

The Future

- Requested new definable tag for Manager Output “CallLabel” that would act just like CallerID to be able to use it for the reason it was meant to be used
- Research and testing into what causes Asterisk Manager Interface pauses at high loads
- Possibly a more robust Manager interface that allows for direct interaction with a channel, such as DTMF collection, from within the Manager API