Product Support Notice

PSN #	PSN020283u

Original publication date: 08-Mar-17. This is Issue #01, published date: 08-Mar-17.

Severity/risk level: Medium | Urgency: When convenient

Name of problem

Avaya Aura® Application Enablement (AE) Services CTI agent client design best practices.

Products affected

Avaya Aura® Application Enablement (AE) Services, Releases 6.3 - 7.0.1

Problem description

This notice provides best practices that should be implemented when developing CTI clients/applications integrating with Application Enablement (AE) Services.

These best practices apply to DMCC Third Party Call Control (3PCC), TSAPI, JTAPI, CVLAN, and ASAI call control and exclude DMCC First Party Call Control. DMCC First Party Call Control is documented in the DMCC Programmer's Guide.

Note that CTI agent application/soft client development experience is assumed.

Resolution

When designing CTI agent client/applications that integrate with AE Services, the following development guidelines should be followed to prevent overloading AE Services and causing AE Services message response latency.

- The application should implement an outstanding query/request (outstanding messages) counter and threshold.
  - The counter should be incremented when a message is sent and decremented when a response to a previously sent message is received.
  - The threshold should be set relatively conservatively (e.g. 10 concurrent outstanding messages). The application should not exceed the outstanding message threshold except in exceptional situations. For example, the application should send out 10 queries/requests incrementing the counter each time, then wait until the counter is less than 10 before sending additional messages.
  - To balance between polling message needs (e.g. agent state: TSAPI cstaQueryAgentState) and other message requests (makeCall: TSAPI cstaMakeCall, Answer: TSAPI cstaAnswerCall, ClearConnection: TSAPI cstaClearConnection), etc.), the application may need to exceed the outstanding message threshold for brief periods of time (to accommodate message bursts), but the application should back off as soon as possible to compensate and bring the message rate back within the outstanding message threshold. The application might also need to prioritize call control messages over agent state query messages to assure the most important messages are sent first.
  - It is desirable to make the threshold configurable so that other variables, such as the number of concurrent CTI applications making queries/requests, can be accounted for and the threshold can be adjusted accordingly.
    - Furthermore, it might be feasible for the application to implement a threshold that is internally and automatically adjusted based on the response time to outstanding message requests. If the AE Services message response time starts slowing the threshold could be adjusted (lowered). Once the threshold is adjusted dynamically and the response time recovers to an acceptable level the threshold could be adjusted accordingly (increased) again. This allows for more dynamic real-time application message throttling adjustments based on AE Services message response latency.
    - Note that specific solution dependent tuning of thresholds might be required on a recurring basis to account for ongoing solution changes (e.g. changes to agent splits, VDNs, CM traffic increases, adding additional applications to the AE Services integration, etc.).
- The application should implement an overall message rate threshold.
  - The overall message rate threshold must be accounted for in addition to the outstanding message threshold described above. The overall message rate threshold controls the amount of messages sent in a given period of time, for example X messages in Y time interval (e.g. X might be 500 and Y might be 1 second for a 500 message/second threshold). Preferably Y should be more granular than one second, such as 100ms intervals.
    - X and Y should be configurable to allow the overall message rate to be changed as necessary.
    - Note that specific solution dependent tuning of X and Y might be required on a recurring basis to account for ongoing solution changes (e.g. changes to agent splits, VDNs, CM traffic increases, adding additional applications to the AE Services integration, etc.).
- The following conceptual example provides a simplistic algorithm that implements the counters and thresholds described above.

While True
If MessagesToSend is greater than 0 And
MessagesSentInInterval is less than MaxMessagesPerInterval And
OutstandingMessages is less than MaxOutstandingMessages
Then
Send the next message
Increment MessagesSentInInterval
Increment OutstandingMessages
Else
Sleep for a small interval

- With the above example *OutstandingMessages* is decremented for every message response received. Also, *MessagesSentInInterval* is reset to 0 after the Y time interval has passed.
- The algorithm can be further refined as needed. For instance, if request prioritization is desirable or necessary (e.g. give call control priority over queries) the algorithm can be modified fairly simply to meet these specific needs. Multiple send queues and priorities attached to send queues can be implemented for example.

- Polling for agent states should follow these general guidelines.
  - Initial agent state polling at startup or after error recovery. This involves populating polling lists with initial agent states at startup and after an error condition occurs in which polling lists can become out of date. The polling lists should be populated in the following order (TSAPI examples provided):
    - Domain control (cstaMonitorDevice()) for all split/skills in which agents reside. This provides login/logout events for all agents.
    - Monitor (cstaMonitorCallsViaDevice()) for all VDNs that route calls to splits/agents. This provides call state information for all agents in most environments. If direct agent calling is utilized, monitors (cstaMonitorDevice()) must be established on those agent-IDs which are direct dialed.
    - Query (attQueryAgentLogin()) for logged in agents in all splits/skills. This provides the current set of logged in agents.
      - Add all logged in agents to an agent state query list. Note that an agent state is exactly that, an agent busy on an ACD call for skill 1 will be busy for all skills, so an agent should never appear more than once in this list regardless of the number of skills into which they are logged in.
  - Agent steady state polling. This involves maintaining agent state lists once the initial state is known.
    - Periodically querying agent states for all agents on the agent state query list. Note that this list will already include all of the agents determined at startup or after error recovery. Agent state queries should be performed based on the message counter and threshold guidance provided earlier.
      - The frequency of querying a specific agent for its state should not be excessive (e.g. no more than once every 10 seconds) and preferably should be configurable.
    - When an event is received indicating that an agent is on a call, remove the agent from the agent state query list and move them to another list e.g. active list). In other words, never poll for agent state when the agent is active on a call.
    - When an event is received indicating that an agent has dropped from the call, add them to the agent state query list to re-determine their agent state.
    - When an agent logout event is received, remove the agent from all lists.
    - When an agent login event is received, add the agent to the agent state query list.
  - Note that a single SMS query can be used to get a list of skills an agent has provisioned when the agent logs in. Another method that has been used is to implement an extra “dummy” skill on CM. The dummy skill is configured on all agents and the application monitors only this dummy skill. Then, only a single event is received when an agent logs in or logs out.
- If the CTI application implementation spans multiple servers, each with its own communication link to AE Services, ensure that the servers do not make redundant requests to AE Services. Either have an architecture where the servers share information acquired from AE Services amongst themselves or segment the solution so each server is responsible for a proper subset of the information to be acquired (VDNs, extensions, agents, etc.).

Workaround or alternative remediation

n/a

Remarks

n/a

Patch Notes

The information in this section concerns the patch, if any, recommended in the Resolution above.

Backup before applying the patch	
n/a	
Download	
n/a	
Patch install instructions	**Service-interrupting?**
n/a	Yes
Verification	
n/a	
Failure	
n/a	
Patch uninstall instructions	
n/a	

Security Notes

The information in this section concerns the security risk, if any, represented by the topic of this PSN.

Security risks
n/a
Avaya Security Vulnerability Classification
Not Susceptible
Mitigation
n/a

If you require further information or assistance please contact your Authorized Service Provider, or visit support.avaya.com. There you can access more product information, chat with an Agent, or open an online Service Request. Support is provided per your warranty or service contract terms unless otherwise specified in the Avaya support Terms of Use.