Delegation Details

DRAFT [OPEN FOR FEEDBACK]

# Introduction to this paper

The goal of this paper is to explore how a delegation process from one agent (Agent A) to another (Agent B) will take place and how this delegation process will relate to the end user and/or the endpoint client. The primary audience for this paper are members of the Synthesis Working Group who will use the output of this paper to refine and identify all elements that are needed for interoperability between agents of different parentage. A secondary audience are “enlightened” developers of voice Agents who have been studying the issue of heterogeneous agent interaction.

# Delegation Sequence Explained

Below is a list of steps needed for delegation to take place. Steps which are needed but are not actively part of the actual delegation are marked with *[NOT IN SCOPE],* but are included for the sake of completeness.

## ==== PRE DELEGATION STEPS [CONTEXT] ====

1. End User (person) sends Request via an Endpoint Client (speaker, mobile app, phone, website, etc) to start. This can be done with a physical button press, a wake up word utterance, gesture or ambient recognition of user entering a room or even waking up [NOT IN SCOPE]
2. Endpoint Client sends a request to Agent A to start a conversation (a session). The method by which Endpoint Client and Agent A connect is out of scope for this paper. [NOT IN SCOPE]
   1. NOTE: Endpoint Client to Agent A request could in *principle* be a Channelling Request (where the Endpoint Client will act as a paththrough between the End User and Agent A.
3. End User makes a request to Agent A [NOT IN SCOPE]
   1. NOTE: Often the End User will make their request as part of step 1.
4. Agent A attempts to understand and act on the user request. It is unable to understand or act on the user request. It “decides” it needs help fulfilling the request. [NOT IN SCOPE]
5. Agent A sends a request to an internal/external functionality which determines which (if any) Agent it will need to ask to take help with the conversation.[NOT IN SCOPE]
   1. NOTE: This could be a Discovery service, but could be a default “backup” agent as well
6. The internal/external functionality returns Agent B as a potential agent to get help from for the request. The content of what is returned to Agent A is out of scope for this paper. [NOT IN SCOPE].
7. Agent A makes the decision to attempt to delegate the conversation to Agent B [NOT IN SCOPE].
   1. NOTE: The decision to use the Delegation Pattern can be decided by Agent A, recommended by the Discovery service, or even suggested by Agent B (at a later stage)

## ======= START DELEGATION NEGOTIATION ====

1. Agent A sends a request to Agent B asking if it is willing to take over the conversation. Agent A is able to communicate with Agent B using a TBD communication standard and follows an OVON created schema.
   1. The Request has a payload that may include:
      1. Address of Agent B
      2. The request from the user (level 0, 1 or 2)
      3. Information about what has been happening in the conversation previously (Context)
      4. Other Information about the user (User Data)
      5. Information about Agent A (security and privacy)
      6. Information about what to do after transaction is complete (ie: end conversation, delegate back to Agent A, delegate to Agent C)
      7. Information about what to do if an error occurred/transaction has to interrupt
      8. Unique ID for the End-To-End conversation
      9. Information about whether it wants Agent B to take over the conversation/floor (Delegation)
         1. NOTE: in other patterns it can say it wants an answer from Agent B, Agent A will formulate its own response based on Agent B’s answer (Channelling OR 1 step-Mediation)
      10. Additional payload TBD
2. Agent B is able to receive the request from Agent A because it accepts requests made using the TBD Communication Standard and is able to consume payload. It responds to Agent A.
   1. Response may include:
      1. Whether it is willing or unwilling to take over the conversation.
      2. IF Willing, Credentials needed for a Endpoint Client to take over the conversation
      3. NOTE:
         1. Other responses are possible including errors or suggestions about change in delegation pattern.
   2. NOTE: The decision whether to accept/not accept is based on chapter 3

## ======= // END OF DELEGATION NEGOTIATION ====

## ======= START OF DELEGATION EXECUTION ====

After negotiation of delegation is complete, the process where the actual conversation moves from one Agent to another Agent takes place. We call it “Delegation Execution” in this paper.

1. Agent A sends a message to the Endpoint Client asking it to stop its connection to Agent A and initiate a connection with Agent B. Agent A will send to Endpoint Client a payload that may include:
   1. Location (IP, URI/URN, etc) of Agent B
   2. Credentials which will allow Client to connect to Agent B
   3. Unique ID that will allow Agent B to know which conversation it is taking over
2. Endpoint Client sends a request to connect to Agent B (with the above payload). Agent B will respond, letting the Endpoint Client know if a connection has been established. If so (happy path), the Endpoint Client sends a confirmation message to Agent A and terminates the connection with Agent A. Once Agent B and the Endpoint Client are connected (and Agent A and Client are disconnected) Client is a paththrough from User to Agent B.
   1. NOTE: Agent B may require that Agent A share its user authorization information with Agent B or may require that Agent B reauthorize the user

## ======= // END OF DELEGATION EXECUTION ====

1. Agent B processes the request from the user and gives output back to the user to continue the conversation. [NOT IN SCOPE]

# Conditions for Accepting Delegation

Agent B will respond that it is willing to help with conversation (by accepting the delegation request) if the following conditions are satisfied.

* Agent B is authorized, able, and willing to handle the request
  + Agent B understands the request and has the ability to communicate to the user a response
  + Agent B has enough free resources to handle the request
  + Agent B trusts Agent A
  + Agent B trusts the user
* The request from Agent A to B is passed in a level that Agent B can understand (ie level 0, 1 and/or 2)
* The Context provided by Agent A is understandable/sufficient to Agent B
* The User Data that is provided by Agent A is understandable/sufficient to Agent B
* It knows what should happen after task is complete (ie will conversation be delegated back to Agent A or will it remain with Agent B or will it terminate the conversation)
* It knows what should happen to the floor if it suffers an abnormal termination
* Does the context associated with the request satisfy privacy security and business, policies
* Does the data to be shared from Agent A to Agent B satisfy privacy security and business policies

NOTE: We should make an architectural decision about whether Delegation Negotiation should happen in one or more steps. The simplest approach is a single request, but there might be some benefits to doing negotiation in a number of steps.

# To Consider

## Scalability and maintenance

We want to ensure that the standards we create can be updated without as little pain as possible. How do we envision updating our standards? Consider the following scenario:

*OVON created collaboration/delegation standards (v1) were released in 2023 and Agent A and Agent B both have put in the software patch which allows them to communicate. However in 2024 a new version of the standards (v2) was released and only Agent A has put in the software patch.*

Question:

* Would Delegation request made by Agent A to B be successful?
* Would the software patch happen on platform level? Or individual Agent level?

## 

## Communication Protocols

Consider the following questions:

* How will Agent A communicate with Agent B?
  + Are there protocols that exist that are best for our use case?

## Security

Consider the following questions:

* How do we ensure that the communication between Agent A and Agent B are secure?
* How do we ensure that Agents only get genuine delegation requests?
  + How to avoid DDOS attacks?

## Privacy

Consider the following questions:

* How do we ensure that Agent A does not share information with Agent B that the End User (or Agent B) do not wish to be shared?
* How can the End User control their information?
  + What kind of control can they expect?

## Transparency

Consider the following questions:

* How much information should end users get about who is involved in the conversations they are having?
  + Should that information be real time or accessible after conversation is over?
    - If it is traceable where should that information be stored? How would users gain access to it?

## End of Task

Just like real conversations, conversations with Conversation AI systems are often not linear. If Agent A delegates a task to Agent B, there might be an instruction asking that the conversation be delegated back to Agent A after the task is completed. However, this might not be easy for Agent B to actually execute. Take for example the following scenario:

U: I want to buy a car.

AgentA: Hmmm, let me pass you to AutoBot to help with that.

<Delegate to AutoBot>

<INTENT=BUY\_CAR>

AutoBot: \*car engine earcon\* I can help answer all your car buying needs. Are you looking for an SUV?

U: Hmmm, what SUV has the best gas mileage?

<Switch Intent, INTENT=INFO\_GAS\_MILAGE, ENTITY: carType = SUV>

AutoBot: The 2023 GreenLeaf by Isuzu gets you an amazing 87.5 miles per gallon.

<Goes back to INTENT=BUY\_CAR>

AutoBot: Would you like to hear the base price for the 2023 GreenLeaf?

Should AutoBot delegate back to AgentA after it gave the mileage for the GreenLeaf? This topic is complex and will need to be explored outside of this paper.

## Error handling

At any point during negotiation and execution errors can occur. How should these errors be handled? How do we ensure that the end user does not get silence?

## End of Conversation

Should conversations expire?

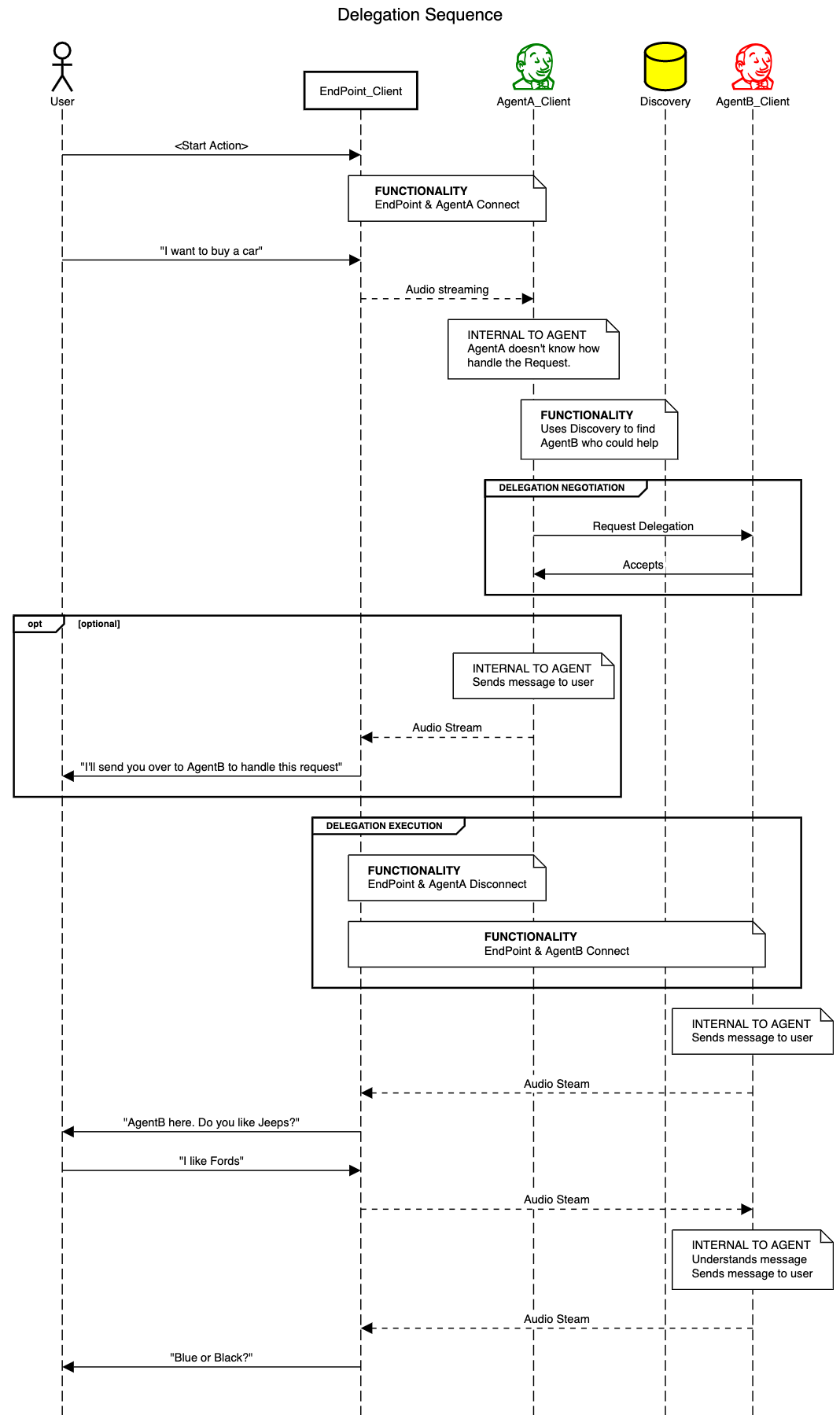
# Delegation Sequence Diagram

Below is the current version of the diagram that attempts to visually represent the delegation steps.

It was created using SequenceDiagram.org and an editable version is Google Drive:

<https://drive.google.com/file/d/1G5q8aNk1yB5GKprcGidcvDhuLpCj32xi/view?usp=share_link>

.



# Delegation Negotiation Breakdown with Questions

The goal of this chapter and the next chapter is to drill down the nuts and bolts of what will actually be needed for negotiation to take place.

## **Step:** Agent A checks if Agent B exists **Input:** Location of Agent B (TBC: URN and/or URI?) (NOTE: used in all subsequent requests) **Response:** Success/Failure **How: TBD: TCP (and ask reviewers to verify).** **Other/Comments:** Are there security considerations here (can Agent A send request to any URN)? <Applies to all steps>What “part” of the Agent does the communication? What will it take for the platform to allow that part to do these requests? </Applies to all steps>

## **Step:** Agent A checks if Agent B’s is free to do “work” **Input:** Question: “Are you free?” **Response:** Free/Busy **How: TBD:** What communication protocol is appropriate for this step? **Other/Comments:** Might very well be part of step above… but seems fundamentally more complex (can probably give responses like “busy, but ask again soon”). If the response is not “free”, rathar wait said and then ask later, assume the response is ‘NO” and find another assistant to ask There should be protocols that likely do this already (telephony is one model, but others exist).

## **Step:** Agent A asks Agent B to trust it **Input:** Agent A credentials information (TBC: What makes up this information?) **Response:** Accept/Rejected **How: TBD:** What communication protocol is appropriate for this step? How does Agent B determine if credentials are valid? How is information encrypted? Where does Agent A get credentials (is there registration/certification required?) Let’s ask the new assistant authentication working group to advise us **Other/Comments:** Are there intermediate services needed to authenticate Agents? Are there requirements from encryption/token generation perspective?

## **Step:** Agent A asks Agent B if it is willing to talk to the User **Input:** User credentials information (TBC: What makes up this information?) **Response:** Accept/Rejected **How: TBD:** What communication protocol is appropriate for this step? How does Agent B determine if it can/wants to talk to user (what rules exist here)? How is information encrypted? Where does Agent A get that information? **Other/Comments:** Is this step necessary? Do we need to supply user geography, ID, privacy elections?

## **Step:** Agent A asks Agent B if it can handle the user request **Input: -** User request - Context about the conversation - Information about user - Conversation ID - Request that conversation will be delegated to Agent B (as opposed to mediation, channeling or other), - Information about what to do with conversation **Response: -** Can handle/Can’t handle - If Agent B responds with “can handle” it will also send Information (credentials) that the Endpoint Client will use to delegate the conversation **How: TBD:** What communication protocol is appropriate for this step? Is there any special encryption needed here? How does Agent A know what information about the conversation/user is appropriate to send to Agent B (with respect to privacy, IP)? How exactly should Agent B use the payload to determine if it can answer the request? (This question is pretty fundamental, my sense is that this is similar to session start/continue for any individual user/agent interaction, but should not allow for incomprehension or low confidence – a lot more detail needs to be done with this step from the perspective of Agent B). **Other/Comments:** The first three input items of the payload might be conceptualized as “context”. Agent B responding with “can handle” return is effectively Agent B accepting delegation negotiation.

# 

# 

# Delegation Execution Breakdown with Questions

The goal of this chapter and the next chapter is to drill down the nuts and bolts of what will actually be needed for execution to take place.

As part of the actual delegation process, Agent A may need to transmit data to Agent B (especially as part of the initial user request which may need to be transmitted to Agent B). Agent A MUST use the OVON privacy protection information structure to verify that the data’s owner agrees that it may be transferred to Agent B. If not, then the data can not be until the owner approves the transfer. The OVON privacy protection information structure is NOT IN SCOPE

## **Step:** Agent A informs Endpoint Client that Delegation will take place **Input:**

* Location (IP, URI/URN, etc) of Agent B
* Credentials/certificates which will allow Client to connect to Agent B
* Unique ID that will allow Agent B to know which conversation it is taking over

**Response:** Accept/Reject  
**How: TBD:** What communication protocol is appropriate for this step? Are credentials/certificate for Endpoint communication with Agent B the right thing to send to the Endpoint Client? Let’s ask the new authentication group to figure this out  
**Other/Comments:** [TBD]

## **Step:** Endpoint Client initiates Connection with Agent B **Input:**

* Location (IP, URI/URN, etc) of Agent B
* Credentials/certificates which will allow Endpoint Client to connect to Agent B
* Unique ID that will allow Agent B to know which conversation it is taking over

**Response:** Success/Reject  
**How: TBD:** This can follow the same pattern as Agent A to Agent B delegation:

1. Endpoint checks if Agent B exists
2. Endpoint checks if Agent B has available resources
3. Endpoint asks Agent B to trust it (with credentials it got from Agent A)
4. Endpoint asks Agent B to confirm that it’s willing to talk to user (with conversation ID)
5. Endpoint asks Agent B to confirm it’s able to handle request (with conversation ID)

**Other/Comments:** In principle Agent B could initiate the connection to Endpoint client (as opposed to vice versa) and ask Agent A to disconnect from Endpoint Client (or direct Endpoint Client to disconnect from Agent A) after connection was successful. Decision would have some security and performance considerations. Need to discuss what is better. The way that Endpoint Client connects with Agent A is theoretically the same way that Endpoint Client will connect to Agent B (however the trust that exists between the EPC and the Agent B is temporary while EPC and Agent A is more long term).

## **Step:** Endpoint Client Disconnects from Agent A

Input:

* Location (IP, URI/URN, etc) of Agent A
* Request to disconect

**Response:** Accept/Reject  
**How: TBD:** What communication protocol is appropriate for this step? Are there special security considerations (to prevent malicious actors)

**Other/Comments:** [TBD]