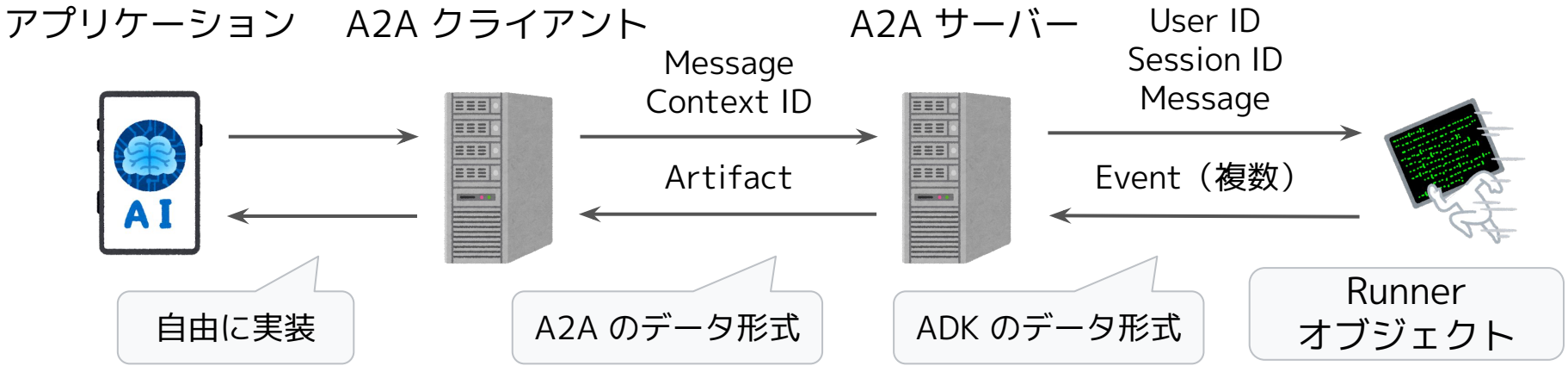
# A2A と ADK を組み合わせる方法

# A2A の役割

- A2A は、エージェント間で送受信するデータのフォーマットや送受信の手続きを定義するもので、送受信するデータの解釈や処理手順は、エージェントの実装に依存します。
  - データの解釈や処理手順がマッチしないエージェント間を A2A で接続しても正しく動作しません。

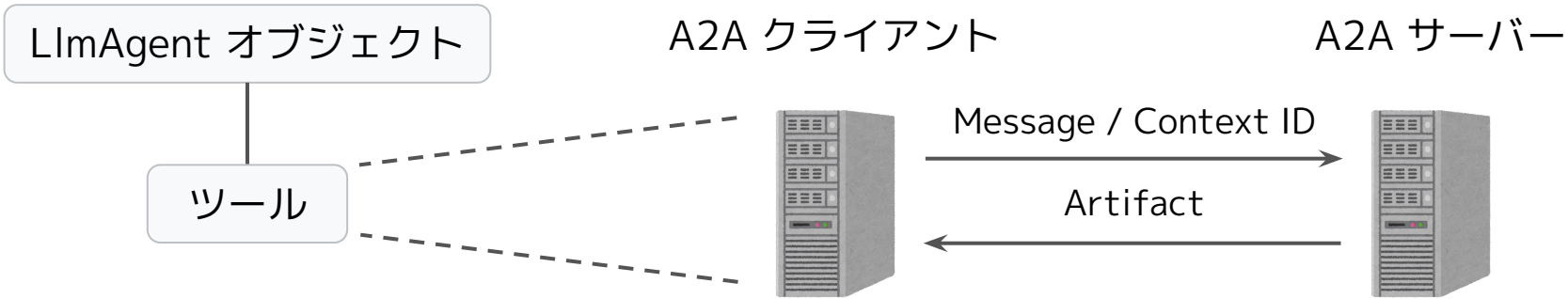
# A2A サーバー経由で ADK のエージェントを利用する方法

- A2A は、実装レベルでは JSON をやりとりする RPC にすぎません。
- JSON でやり取りするデータ形式が決まっているので、A2A のデータ形式と ADK のデータ形式を相互変換すれば、A2A サーバーの裏にある Runner オブジェクトを通じて ADK のエージェントが利用できます。



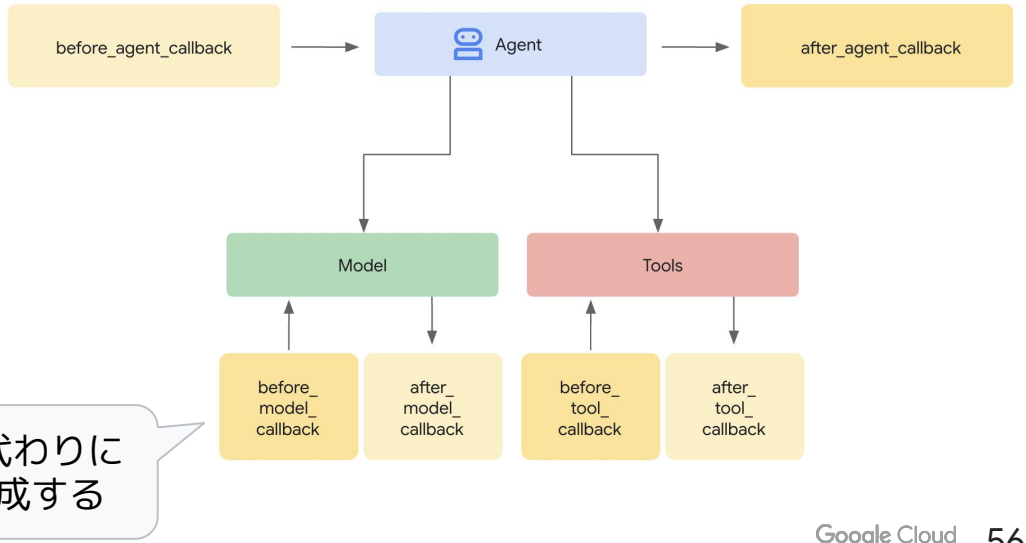
# ADK のエージェントから A2A サーバーを利用する方法

- A2A クライアントとして A2A サーバーとデータをやり取りする関数を実装します。
- この関数を ADK のエージェントに「ツール」として登録すると、必要に応じて A2A サーバーが提供する機能を利用します。



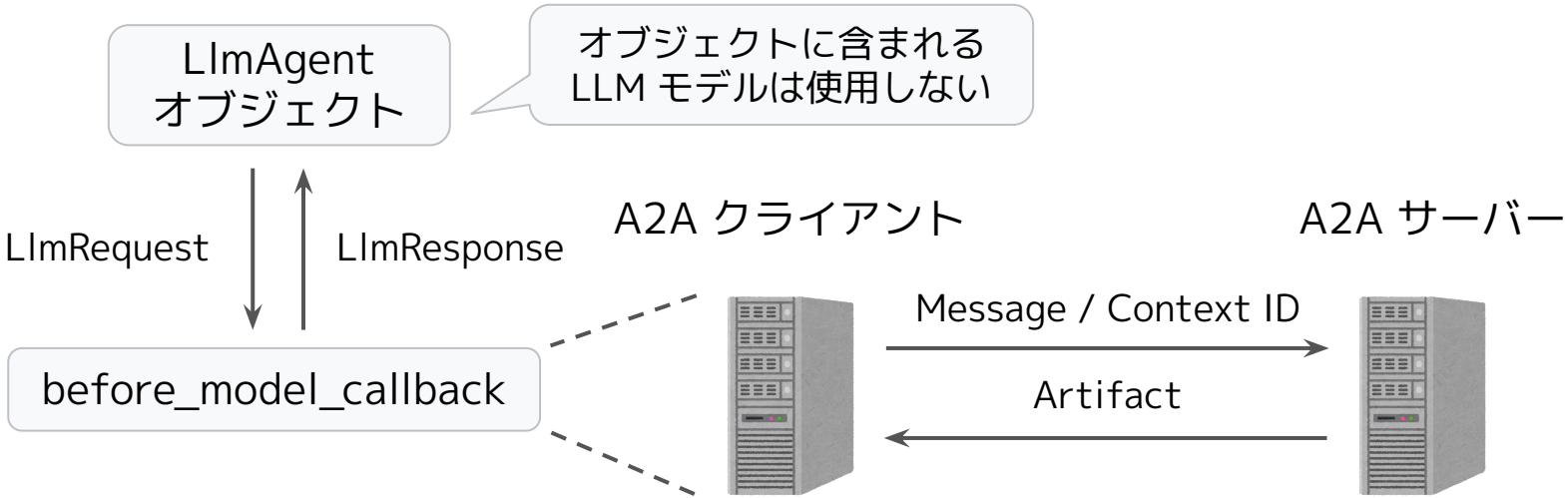
# ADK のエージェントとして A2A サーバーを利用する方法

- A2A サーバー（を通して提供される機能）を ADK のエージェントであるかのように見せる場合は、ADK のコールバック機能を利用します。
  - 「before_model_callback」を利用すると、モデルが受け取るメッセージを横取りして、モデルの代わりに応答を生成できます。
  - このコールバック関数の中で、A2A サーバーにメッセージを送信して、得られた結果をモデルの応答として返却します。



# ADK のエージェントとして A2A サーバーを利用する方法

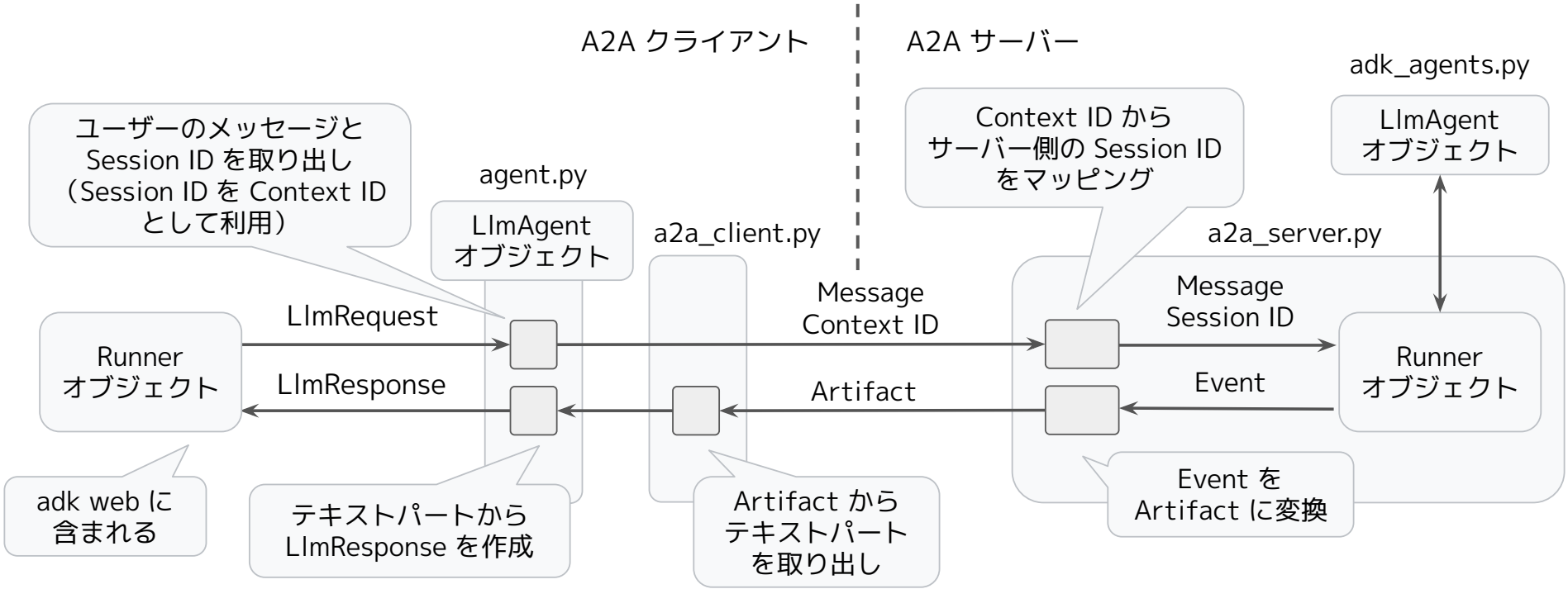
- A2A サーバー（を通して提供される機能）を ADK のエージェントであるかのように見せる場合は、ADK のコールバック機能を利用します。
  - 「before_model_callback」を利用すると、モデルが受け取るメッセージを横取りして、モデルの代わりに応答を生成できます。



# ADK のエージェントとして A2A サーバーを利用する方法

- A2A サーバーとはセッション情報が共有されない点に注意が必要です。セッション情報を共有するには次のような方法があります。
- LlmRequest オブジェクトに含まれる会話履歴をすべて A2A サーバーに送信
  - A2A サーバーの裏側のエージェントは、メッセージに含まれる会話履歴を参照して応答するように実装しておきます。
- ADK の Session ID を A2A の Context ID に変換して A2A サーバーに送信
  - Context ID は Session ID に相当する A2A のオプションメッセージです。
  - A2A サーバーの裏側のエージェントは、Context ID を用いて過去の会話履歴を管理するように実装しておきます。（この場合、クライアント側で保持する会話履歴と A2A サーバー側で保持する会話履歴が一致しない可能性がある点に注意が必要です。）

# A2A サーバーで動作する ADK のエージェントを リモートの ADK から利用する構成例




https://github.com/google-cloud-japan/sa-ml-workshop/blob/main/blog/adk_a2a_integration/README.md