**Transport**

JSON-RPC messages are transmitted between the server and the client. The MCP protocol messages are converted into JSON-RPC for transmission on the transport layer. MCP uses JSON-RPC to encode messages. JSON-RPC messages **MUST** be UTF-8 encoded

JSON-RPC (stateless protocol) can be transported through HTTP, web-socket or TCP. The request from client goes to remote server and then remote server will send back the response, and this communication would seem like the server is a local server, no any remote server.

If no JSON-RPC, then client and server must agree on another way for sending messages. For example, suppose the client send a message: {name:”id-2”} where-else the server knows it like {id:”id-2”}, so this will cause problem.

JSON-RPC gives a standard way (so no incompatibility between server and client)

JSON-RPC is about the structure, not about how the messages are transported.

REQUEST: {

jsonrpc: "2.0",

id: number | string,

method: string,

params?: object

}

RESPONSE: {

jsonrpc: "2.0",

id: number | string,

result?: object,

error?: {

code: number,

message: string,

data?: unknown

}

}

NOTIFICATION: {

jsonrpc: "2.0",

method: string,

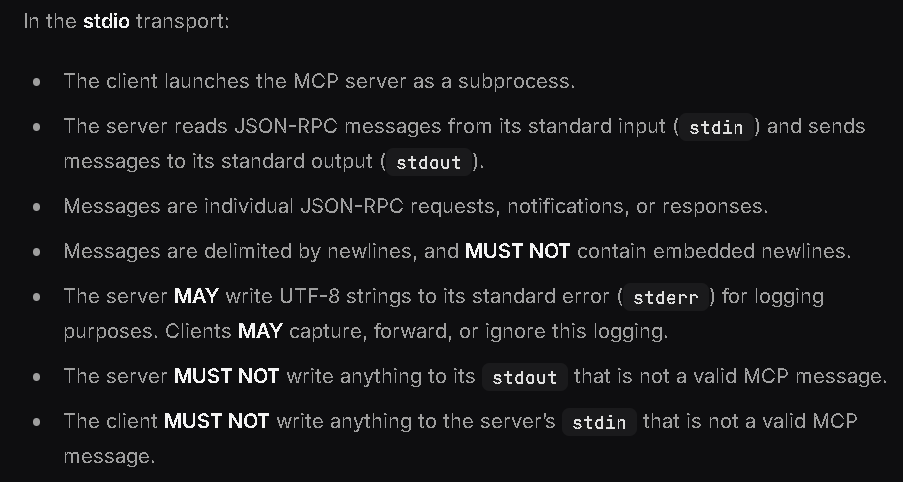
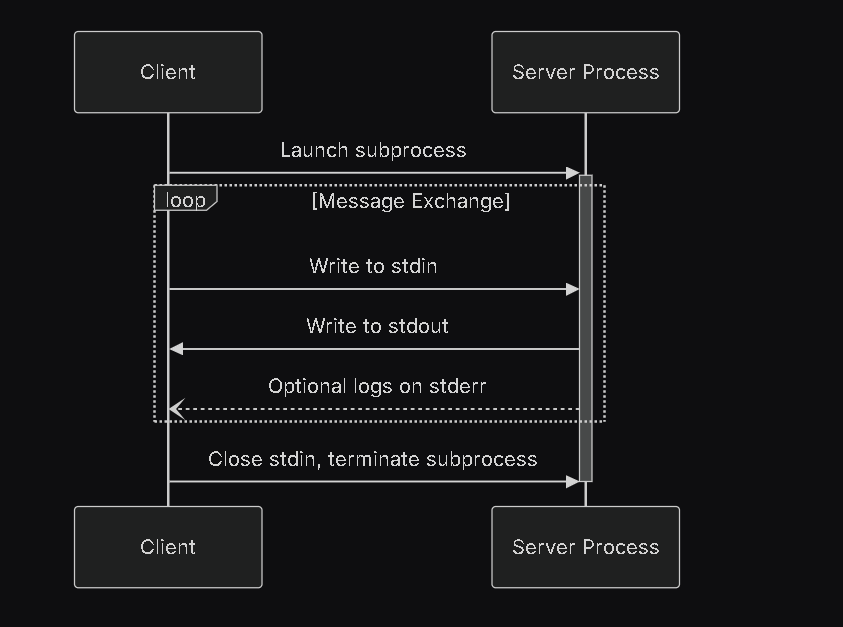
params?: object

}

Lifecycle:

After the client has send an initialization request to server and server has also respond that back, a successful initialization has happened. After successful initialization, the client **MUST** send an initialized notification to indicate it is ready to begin normal operations. **Notifications** are one-way messages that don’t expect a response.

STDIO

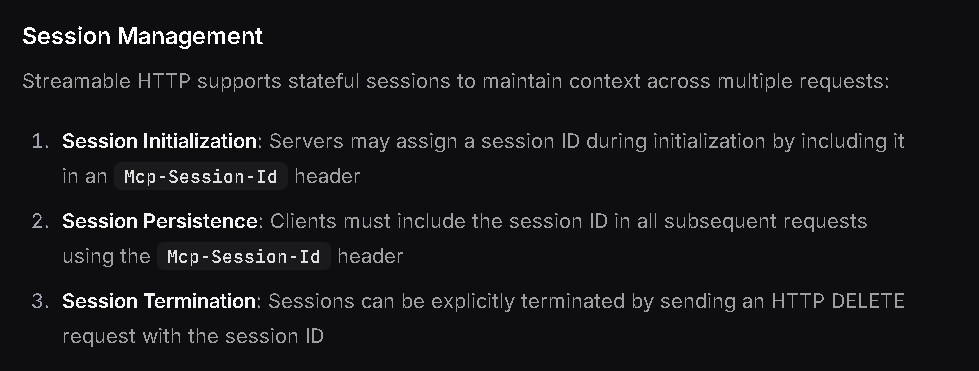
1. Server is local.
2. 
3. 
4. If we use any MCP server deployed in npx, uvx package or deployed on github or any server created by other, if it is an stdio server, then when we use the config.json file or created client for it and run that, these servers are downloaded into our computer and then they run locally. (These locally running servers have security concerns like someone’s stdio server that we use and download locally into our computers, can access any file system and can do sth with that)
5. But the streamble-HTTP servers can be remote servers which do no needed to be installed or downloaded locally. These remote servers can run as API exposed with the authentication for the server. These APIs can be used by agents (ie. <https://gofastmcp.com/integrations/openai> ).

HTTP-STREAMABLE

1. Server is local or remote.
2. This replaces the legacy HTTP+SEE
3. In the **Streamable HTTP** transport, the server operates as an independent process that can handle multiple client connections.
4. The server **MUST** provide a single HTTP endpoint path (hereafter referred to as the **MCP endpoint**) that supports both POST and GET methods. For example, this could be a URL like <https://example.com/mcp>.

**How it works?**

1. Client-to-Server Communication: Every JSON-RPC message from client to server is sent as a new HTTP POST request to the MCP endpoint
2. Server Responses: The server can respond either with:
3. A single JSON response (Content-Type: application/json)
4. An SSE stream (Content-Type: text/event-stream) for multiple messages
5. So, client sends a POST HTTP request to the server and server sends back either JSON response or SSE Stream response.
6. Server-to-Client Communication: (Optional: Support server-initiated SSE streams) Servers can send requests/notifications to clients via:
7. SSE streams initiated by client requests
8. SSE streams from HTTP GET requests to the MCP endpoint



Session ID is more important where streaming or SSE Streaming happening as it needs to maintain a long-connection. It is more important for stateful connection.

