

Great question — and you're touching on a core design principle of MCP (Model Context Protocol).

🔍 “Why do I have to specify which resources to load (e.g., `uris=["greeting://Alice"]`) — can't it just load *all* resources automatically like it does for tools?”

Let's break this down clearly.

✅ **Short Answer:**

Tools can be listed fully → so `load_mcp_tools()` loads *all*.

Resources *cannot* be fully listed in a useful way → so you must request *specific URIs*.

Here's why.

🔍 **1. Tools: Finite and Discoverable**

- Every MCP server supports the `tools/list` method.
- It returns a complete list of all available tools with:
 - Name
 - Description
 - Parameters
- Example:



json

```
[  
  { "name": "sum_numbers", "description": "Adds two numbers" },  
  { "name": "get_weather_alerts", ... }  
]
```

✅ So `load_mcp_tools()` can safely fetch everything — it's a closed set.


2. Resources: Two Types — Only One Is Fully Listable

MCP defines two kinds of resources:

Direct Resources(fixed URIs)	 Yes — viaresources/list	file://config.json,calendar://today
Resource Templates(dynamic)	 No — infinite possibilities	weather://forecast/{city},user://{id}/profile

The Problem:

- Your server might expose a template like person://{name}/age.
- There's no way to list all possible values of {name} — it could be *any string*!
- So resources/list only returns fixed, direct resources — not templates.

 You cannot “load all resources” because most useful resources are dynamic and unbounded.

Why You Must Specify URIs

When you do:

```
await load_mcp_resources(session, uris=["person://ali/age"])
```

You're saying:







“I know this specific resource exists — go fetch its current value.”

This is by design:

- Resources are data, not functions.
- The AI (or app) must know what it wants before reading it.

- It's like opening a file: you need to know the path (/etc/config) — you can't "load all files on the internet".

Comparison: Tools vs. Resources




Full discovery	 tools/list → complete list	 resources/list → only fixed URIs
Dynamic usage	 Fixed set	 Infinite via templates (//user/123)
Load "all"?	 Yes — safe and useful	 No — impossible or meaningless
How LangChain loads them	load_mcp_tools() → all	load_mcp_resources(uris=[...]) → specific

Real-World Analogy

- Tools = Menu items at a restaurant
→ The waiter can hand you the full menu (tools/list).
- Resources = Custom orders or live data
→ You can't say "give me all possible dishes ever" — you must say:
"I want the weather in Paris" → weather://forecast/paris
"I want Ali's age" → person://ali/age

The system can't guess what dynamic data you want.

Best Practice

-  Use load_mcp_tools() → get all tools (safe, complete).
-  Use load_mcp_resources(uris=[...]) → request only the resources you need.
-  If you have fixed resources, you can list them first:

```
direct_resources = await session.list_resources() # gets fixed URIs
```

```
data = await load_mcp_resources(session, uris=[r.uri for r in direct_resources])
```

But this won't include templates like `person://ali/age` — you must know those URIs in advance (from docs, prompts, or AI reasoning).

Summary

You can't load all resources because most are dynamic and infinite.

You can load all tools because they're a finite, predefined set.

So yes — you must specify resource URIs. That's not a limitation; it's how MCP enables scalable, parameterized data access.

Great question — you've grasped a subtle but important distinction! 🙌