AURAPLAYER
Innovation Without Disruption

TECHNICAL DATASHEET
FOR ARCHITECTS & IT TEAMS

AURAPLAYER MCP SERVER: DEPLOYMENT ARCHITECTURE, PREREQUISITES & OPTIONS

CONNECTING AI AGENTS TO ORACLE EBS, YOUR WAY

A technical reference for architects and IT teams evaluating how to connect AI agents to Oracle EBS through the AuraPlayer MCP server — covering both the local (stdio) and enterprise (Streamable HTTP) deployment models, authentication, network exposure, and rollout.

Governed, process-layer MCP access to Oracle EBS
for the AI tools your enterprise already uses

The Oracle MCP Landscape: Where AuraPlayer Fits

There's more than one way to skin a cat, and the same is true for connecting AI to Oracle. Most approaches connect an AI agent straight into the database, which is fast to stand up but puts table-level read/write access one prompt away, with no guarantee that Oracle's own validations, approvals, and workflows are respected along the way. AuraPlayer takes a different route: it runs predefined, secured, and governed services that automate Oracle E-Business Suite processes at the application layer, so every trigger, validation, and approval your team already built keeps doing its job.

Worth noting: AI clients such as Claude, Microsoft Copilot, ChatGPT, and Gemini aren't MCP servers at all. They're the interface that discovers and calls whichever MCP server below fits the task, so any of them can be pointed at any of the four options.

Comparing the Four Ways to Connect AI to Oracle EBS

	Oracle SQLcl MCP	Oracle Database Tools MCP	Oracle ORDS Streaming MCP	AuraPlayer MCP
Layer & what it exposes	Database layer. Local, stdio-based CLI server; credentials and connections stay on the developer's machine	Database layer. Oracle-managed, cloud-hosted MCP server for OCI databases	Database layer. REST/ HTTP MCP server exposing the Oracle Database over standard HTTPS	Business process layer. Predefined, governed Oracle EBS workflows, not tables
Key benefit	Nothing to deploy beyond SQLcl; ideal for local SQL generation, schema exploration, AWR analysis	Centrally managed, enterprise-scale MCP access to OCI databases; no per-machine install	Modern, standards-based DB connectivity for SaaS AI clients, no custom integration	Existing Forms validations, approvals, and audit trails run exactly as they do today. Nothing bypassed
Key risk	Runs under the developer's own DB privileges; blast radius depends entirely on that account's grants	Still direct database access; governance depends on OCI IAM policy, not application-layer rules	Still a direct database connection; validations above the DB aren't enforced	The target process must be recorded and published as a service first
Best for	Individual developers and DBAs working locally against a dev or test database	Cloud teams centralizing MCP access and governance across OCI databases	Giving AI clients broad, secure query access to database content	Letting AI safely execute EBS transactions: approvals, receipts, orders, expenses
Example ask	"Explain this AWR report"	"Compare schemas across these two OCI databases"	"List all overdue invoices over $100k"	"Approve every invoice that matches our policy"

The take-away for architects

- **These aren't competing options, they're different responsibilities.** The three Oracle database MCP servers answer questions. AuraPlayer executes business processes.
- **The strongest architectures combine both:** query insight from the database layer, then governed action through AuraPlayer, orchestrated by whichever AI client your team already uses.
- **Ask the right question first.** Not "which MCP server is better," but "what am I asking AI to do?" The answer decides the layer.

Introducing AuraPlayer: Two Deployment Models. One MCP Server.

The AuraPlayer MCP server publishes your recorded Oracle EBS workflows as tools that AI agents can discover and call. It runs in two deployment modes. Both drive the same AuraPlayer Server (ServiceManager) and the same Oracle EBS instance. The difference is where the MCP server lives, how clients reach it, and how identity is handled.

OPTION 1

LOCAL DEPLOYMENT

stdio transport. A lightweight MCP bridge installed on each workstation. Runs under each user's own credentials. Ideal for pilots and desktop clients.

OPTION 2

ENTERPRISE DEPLOYMENT

Streamable HTTP transport. One shared, stateless service for every user and every AI client. Central auth, audit, and control. Nothing installed on client machines.

Use the comparison below to pick a starting point. The pages that follow detail the architecture, prerequisites, and assumptions for each option.

At-a-glance comparison

Aspect	Local (stdio)	Enterprise (Streamable HTTP)
Where it runs	On every user workstation, launched by the MCP client	One shared service next to the AuraPlayer Server (VM, container, or K8s)
Transport	MCP stdio (JSON-RPC over stdin/stdout)	MCP Streamable HTTP: a single `/mcp` endpoint, responses may stream as SSE
Install footprint	Per-machine install, configure, and update (.mcpb bundle or exe)	Zero client install. Clients are pointed at a URL
User identity	Built in. Each install carries that user's AuraPlayer/EBS credentials	OAuth 2.1 via your corporate IdP, with per-user impersonation into EBS
Network reach	Every workstation needs direct access to the AuraPlayer Server	Only the MCP endpoint is published. The AuraPlayer Server stays internal
Client support	Desktop MCP clients that can launch local processes (e.g. Claude Desktop). Web clients cannot use it	Any MCP client supporting Streamable HTTP: claude.ai, ChatGPT, Copilot Studio, Claude Code, IDEs, custom agents
Central control	None. No single place to audit, rotate credentials, or enforce policy	Upgrades, auditing, service filtering, and credential policy handled in one place
Best for	Individual pilots, demos, small technical teams	Production rollouts, SaaS assistant connectors, governed enterprise access

What stays the same in both models

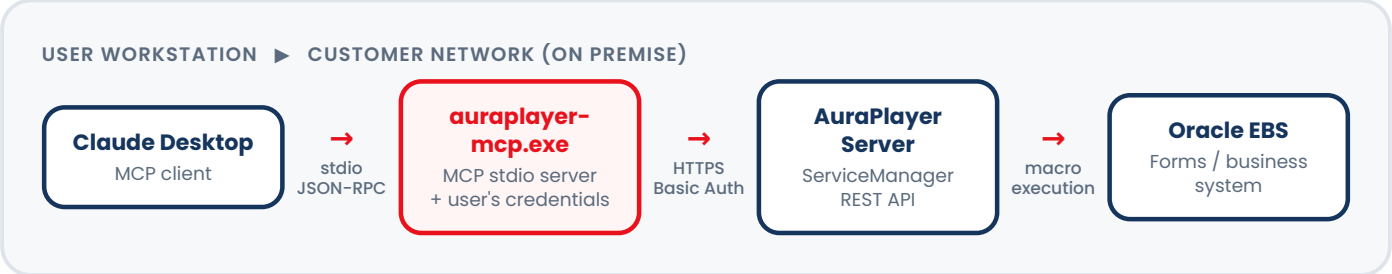
- **The business layer.** Tool calls execute recorded macros through the ServiceManager REST API, so Oracle Forms triggers and validations always run.
- **Tool discovery.** Services labeled "MCP" in ServiceManager are published automatically as MCP tool definitions.
- **Oracle EBS is never exposed.** In both models, EBS is reached only through the AuraPlayer Server inside your network.

Option 1: Local Deployment (stdio)

The MCP server is a small Python bridge, packaged as a standalone Windows executable and installed on every client machine. The MCP client, for example Claude Desktop, launches it as a child process and communicates over stdin/stdout using the MCP stdio transport.

Each installation is configured with that user's AuraPlayer/EBS credentials through environment variables. These are entered in the Claude Desktop extension UI and stored in Windows Credential Manager.

Architecture



Components

Component	Location	Role
MCP client (Claude Desktop, etc.)	User workstation	Hosts the conversation, decides which tools to call
auraplayer-mcp stdio server	User workstation	Translates MCP `tools/list` / `tools/call` into AuraPlayer REST calls
AuraPlayer Server (ServiceManager)	Customer network	Publishes recorded macros as services and executes them
Oracle EBS	Customer network	The business system the macros drive

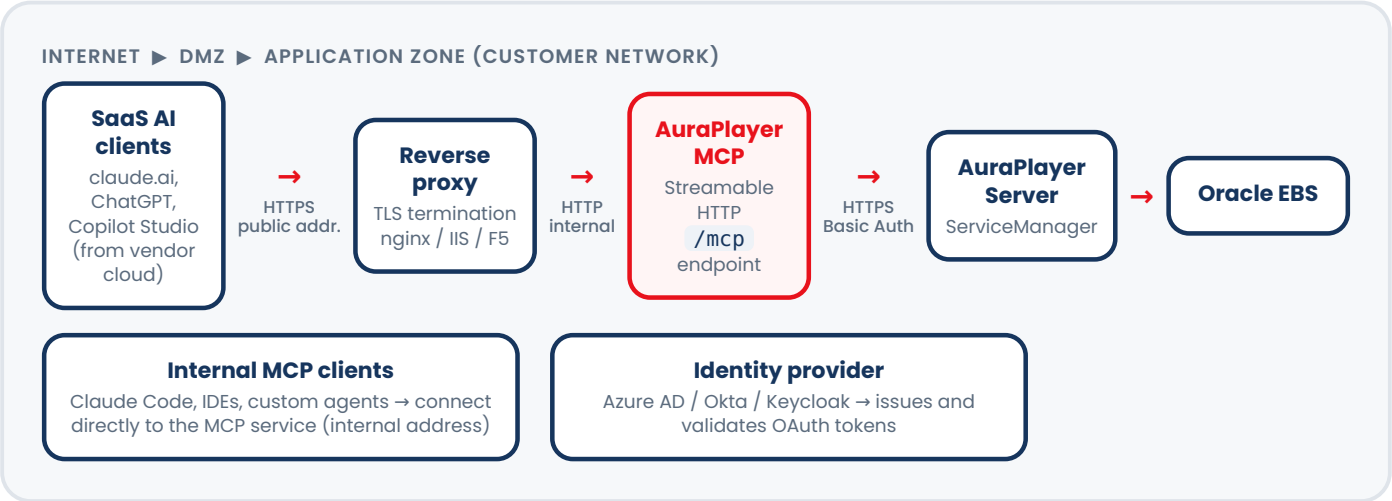
Properties and limitations

- **Per-machine install.** The .mcpb bundle or exe must be installed, configured, and updated on every workstation.
- **Per-user identity comes for free.** Each install carries that user's credentials, so macros run under the correct EBS identity.
- **Network reach.** Every workstation needs direct network access to the AuraPlayer Server.
- **Client support.** Limited to MCP clients that can launch local stdio servers (desktop applications). Web clients such as claude.ai cannot use it.
- **No central control.** No single place to audit usage, rotate credentials, enforce policy, or roll out new versions.

Option 2: Enterprise Deployment (Streamable HTTP)

In the enterprise deployment, the MCP server runs as a single shared service next to the AuraPlayer Server, exposing the MCP Streamable HTTP transport. There is one `/mcp` endpoint. Requests are HTTP POSTs, and responses may stream as Server-Sent Events. Nothing is installed on client machines. Clients are simply pointed at a URL.

Architecture



Key points

- **Flexible placement.** The MCP service is typically co-located with the AuraPlayer Server, but it is a stateless HTTP service whose only requirement is HTTP(S) access to the ServiceManager REST API. It can run on the same machine, a VM, or a container. Co-location keeps latency low and avoids another firewall path.
- **The AuraPlayer Server is never exposed.** Only the MCP `/mcp` endpoint is published, and only as far as needed.
- **Ports.** The MCP service listens on a configurable port (default 8000, plain HTTP, distinct from ServiceManager's default 8080). Clients never see that port. They connect to the reverse proxy on standard 443, which forwards to the MCP port. The only network paths needed are proxy → MCP port, and MCP host → ServiceManager port.
- **One deployment serves everyone.** All users and all client types through one endpoint, with upgrades, auditing, service filtering, and credential policy handled in one place.

Service endpoints

Endpoint	Purpose
`POST /mcp`	The MCP Streamable HTTP endpoint used by all clients
`GET /health`	Unauthenticated health check for monitoring and load balancers
`GET /.well-known/oauth-protected-resource`	OAuth resource metadata for connector discovery (OAuth mode only)

The service is stateless, so it can be restarted or load-balanced freely. Listen address and port, authentication mode, API keys, identity provider settings, and the impersonation user field are all set through configuration.

MCP Client Compatibility

MCP is an open standard adopted well beyond Anthropic. One Streamable HTTP endpoint makes AuraPlayer services available across the AI tools your enterprise already uses. No client needs custom development. The notes below are configuration steps on the client side.

Client transport matrix

Client	stdio (local)	Streamable HTTP (remote)	Requirements / notes
Anthropic claude.ai (web), Claude mobile	No	Yes (custom connector)	Connects from Anthropic's cloud. Needs a public HTTPS address; OAuth or open endpoint. All plans
Anthropic Claude Desktop	Yes (.mcpb)	Yes (custom connector)	stdio for the local install. Custom connectors connect from Anthropic's cloud, not the desktop
Claude Code, Claude Agent SDK	Yes	Yes	Connects directly from the user's machine (internal address is fine). Supports bearer-token headers and OAuth
OpenAI ChatGPT	No	Yes (also SSE)	Custom connectors require Developer mode (beta). Write-capable tools only on Business / Enterprise / Edu; Plus/Pro read-only. Connects from OpenAI's cloud
OpenAI Agents SDK, Responses API remote MCP	Yes (Agents SDK)	Yes	Responses API "remote MCP" is HTTP-only and connects from OpenAI's cloud
Microsoft Copilot Studio, Microsoft 365 Copilot	No	Yes (HTTP only)	Registered through a Power Platform custom connector (auto-generated). Connects from Microsoft's cloud
VS Code / GitHub Copilot agent mode	Yes	Yes	Connects directly from the user's machine
Google Gemini CLI, Agent Development Kit	Yes	Yes (also SSE)	Connects directly. Supports auth headers
Google Gemini Enterprise	No	Yes (HTTP only)	Registered as a custom MCP connector. Connects from Google's cloud
Cursor, Windsurf, JetBrains AI Assistant	Yes	Yes	Connect directly from the user's machine
Custom in-house agents (LangChain, etc.)	Yes	Yes	Framework MCP adapters support both transports

Takeaways

- Every SaaS assistant (claude.ai, ChatGPT, Copilot Studio, Gemini Enterprise) is Streamable HTTP only and connects from the vendor's cloud. This drives the exposure decision (next page).
- Clients running on user machines (Claude Code, VS Code, Cursor, CLI tools) support both transports and can reach an internal address directly.
- A stdio-only client can still reach the shared HTTP endpoint through a local stdio-to-HTTP proxy (e.g. mcp-remote) if ever needed.

Network Exposure & TLS Requirements

The AuraPlayer Server itself never needs an external address. It stays internal in all scenarios. Only the MCP endpoint is published, and whether that endpoint needs a public address depends on which clients must reach it.

Where do connections come from?

Clients	Connection origin	MCP endpoint address
Claude Code, VS Code / Copilot agent mode, IDE plugins, custom in-house agents	The user's machine on the corporate network or VPN	Internal DNS name is enough (e.g. `https://mcp.corp.local/mcp`)
SaaS assistants: claude.ai / Claude Desktop connectors, ChatGPT connectors, Copilot Studio	The vendor's cloud (Anthropic, OpenAI, Microsoft), not the user's machine	Must be reachable from the internet (public DNS + public certificate), typically via a reverse proxy in the DMZ

Two exposure profiles

PROFILE A

INTERNAL-ONLY

No external exposure at all. Users on the corporate network or VPN use clients that connect directly (Claude Code, IDEs, in-house agents). SaaS assistant connectors are not available in this profile.

PROFILE B

EXTERNALLY PUBLISHED

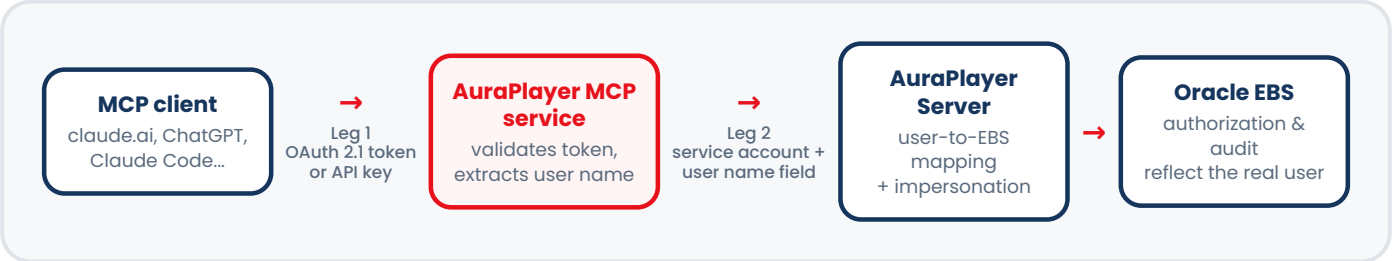
The `/mcp` endpoint is published through a DMZ reverse proxy so vendor clouds can reach it. This enables claude.ai, Claude Desktop and mobile, ChatGPT, and Copilot Studio connectors. Everything behind the proxy stays internal. Vendor clouds publish IP ranges that can be allowlisted at the proxy for defense in depth.

Does it have to be SSL enabled? Effectively yes.

- **Externally published:** HTTPS with a certificate from a public CA is mandatory. Claude, ChatGPT, and Copilot Studio connectors all require a valid `https://` URL.
- **Internal-only:** HTTPS is still strongly recommended because credentials and business data flow over this channel. An internal or enterprise CA certificate is fine.
- **Recommended pattern:** terminate TLS at the reverse proxy and keep the Python service itself plain HTTP on localhost or the app zone. The MCP service then needs no certificate handling at all. Uvicorn can serve TLS directly if a proxy-less deployment is required.

Authentication: The Two-Leg Model

Enterprise authentication has two independent legs. Leg 1 secures the path from the AI client to the MCP service. Leg 2 secures the path from the MCP service into AuraPlayer and Oracle EBS, carrying the real user's identity all the way through.



Leg 1: client to MCP service

The MCP specification standardizes on OAuth 2.1. The MCP service acts as an OAuth resource server that validates bearer tokens issued by your corporate IdP. The SaaS assistants all implement this flow: the user clicks "connect", signs in via the IdP, and the vendor's cloud presents the token on every request. Two practical levels:

Level	How it works	Scope
Static bearer token / API key (simplest)	The reverse proxy or an app middleware checks an `Authorization: Bearer` header	Works for Claude Code, IDE clients, custom agents. Not sufficient for SaaS connectors. Identifies a machine, not a user, so it cannot drive per-user impersonation
OAuth 2.1 against the corporate IdP (full enterprise)	The service publishes OAuth resource metadata pointing at the IdP and validates JWT access tokens on every request	SSO, per-user identity, token expiry and revocation. Works with all connector platforms

Leg 2: MCP service to AuraPlayer (impersonation)

1. Leg 1 authentication yields the corporate user name, for example the JWT's `preferred_username` or email claim.
2. On every service execution, the MCP service injects that user name into a configured input field (e.g. `mcpUserName`). The MCP service itself authenticates to the AuraPlayer Server with a single service account.
3. ServiceManager maps that user name to an EBS user and executes the macro impersonating the mapped EBS user. EBS authorization and audit therefore reflect the real user.

Security properties of this design

- **The user field is owned by the server.** It is hidden from the tool definitions clients see and always overwritten with the authenticated identity, so a client or model cannot spoof another user.
- **The service-account credential exists only on the MCP host.** Never on user machines.
- **Unmapped users are rejected** by ServiceManager, giving a central enable/disable switch per u

Prerequisites, Assumptions & Suggested Rollout

Prerequisites checklist

Requirement	Local (stdio)	Enterprise (Streamable HTTP)
AuraPlayer Server	ServiceManager running, reachable over HTTP(S), with your EBS workflows recorded and labeled "MCP"	
Client machines	Windows workstations with the .mcpb bundle or exe installed, plus network access to the AuraPlayer Server	None. Any MCP client that supports Streamable HTTP
Server host	Not required	A host with HTTP(S) access to ServiceManager: same machine, VM, container, or Kubernetes. Default listen port 8000
Credentials	Each user's AuraPlayer/EBS credentials, stored in Windows Credential Manager	One service account for the MCP service, plus a user-to-EBS mapping table in ServiceManager
Identity provider	Not required	For full enterprise auth: Azure AD, Okta, or Keycloak issuing OAuth 2.1 tokens. API keys suffice for internal pilots
TLS certificate	Handled by AuraPlayer Server config	Internal or enterprise CA for internal-only. Public CA certificate for SaaS connectors
DMZ / reverse proxy	Not required	Only for SaaS connectors: nginx, IIS, or F5 terminating TLS on 443 and forwarding to the MCP port

Key assumptions

- The AuraPlayer Server is never exposed to the internet. It is reachable only from the MCP host and admin machines.
- The MCP service is treated as a trusted subsystem. Protect the service-account credential and the MCP host accordingly.
- Per-user impersonation requires OAuth. API keys identify machines, not users.

Suggested rollout: three phases

1. **Internal pilot**
   HTTP transport + API-key auth + service account. Internal address only, TLS via internal CA. Validates the transport with Claude Code and internal clients. No per-user impersonation yet.

2. **Enterprise auth and impersonation**
   OAuth 2.1 against the corporate IdP. User-name injection into the configured input field. User-to-EBS mapping and impersonation in ServiceManager, plus structured audit logging.

3. **External publication (optional)**
   DMZ reverse proxy, public DNS and certificate. Enables the SaaS connectors: claude.ai, Claude Desktop and mobile, ChatGPT, and Copilot Studio.

About **AuraPlayer**

AuraPlayer's patented technology automatically generates REST APIs, mobile applications, RPA automations, and MCP tools from existing back-office applications, helping you begin your AI transformation while leveraging existing investments.

AuraPlayer's MCP server is built specifically for Oracle Forms and E-Business Suite. It exposes governed business workflows at the process layer, so every AI-initiated action runs through your existing Oracle validations, triggers, and security. No replatforming. No direct database access. Just safe, auditable AI automation on the system you already run.

Let us join your Oracle AI journey.

LinkedIn: @auraplayer

X: @auraplayer

YouTube: @AuraplayerAdmin

Email: info@auraplayer.com

Web: www.auraplayer.com

Schedule a call