Trace a Nexo custom-item display failure from pack to client

A Nexo item can exist on the server while the client displays the wrong model. Trace one item through its definition, generated pack and delivery path. This keeps a broken appearance separate from permissions, recipes or other features that might still work.

Match the documented version family

Nexo maintains an explicit version-support table. For example, the table checked for this guide lists Minecraft 1.21.11 with Nexo 1.16 or newer, and Minecraft 26.2 with Nexo 1.25 or newer. Recheck that table for the selected release rather than treating a minimum from one Minecraft version as support for another. The documentation also marks some intermediate Minecraft releases unsupported.

Record your server, Nexo and test-client versions. Preserve existing Nexo configuration, item definitions, pack inputs and generated output. Choose one affected item whose internal ID you know, and keep an established copy in a private test chest. Retain IDs and persistent data while diagnosing appearance.

Follow the item reference

1. Find the item's YAML definition and identify which appearance mechanism that release uses. Nexo documents item-model definitions alongside modern item components; older examples can use a different arrangement.
2. Check the referenced namespace, path and filename case. Confirm the target texture or model exists in the pack inputs.

3. Read the pack-generation log for the first relevant error. Correct a missing file or malformed model on a copy before changing delivery settings.
4. Generate the test pack using the installed release's documented command. Preserve the previous generated archive so you can compare the outputs.

Inspect merged pack inputs

Nexo's documented pack structure includes `plugins/Nexo/pack/assets/` and `plugins/Nexo/pack/external_packs/`. Imported packs can contain overlapping model or texture files. Check whether an external pack supplies the same path as your intended asset. Test one suspect imported pack on a copy; keep a record of its original position and do not remove unrelated production assets without a plan.

If obfuscation or pack optimisation runs, retain the original inputs and read the processing errors. Do not hand-edit the generated output as your lasting fix: a subsequent build can replace those edits. Correct the source model or build configuration instead.

Confirm delivery and client loading

Review the configured `Pack.server.type` in the release's generated settings. The documented `NONE` mode sends no pack, so it only makes sense where another intended sender handles delivery. A self-hosted mode needs a reachable public address and an allocated TCP port; choose your service's allocation rather than copying an example port.

Join with a matching Java client, accept the pack and examine the client log if loading fails. Test the held item and its inventory icon, then compare an existing copy with a fresh administrative test copy. Protocol translation and Bedrock rendering need separate compatibility checks.

Recover without changing inventories

Restore the previous pack input and delivery configuration if the test regresses other items. Keep item IDs intact. If you need support, send the version combination, one redacted item definition, model path and first generation or client error. Do not include purchased asset archives or private hosting credentials.

Related CloudExa guides

check Java resource-pack delivery · back up the server files · collect the relevant Minecraft logs

Official references

- Nexo server-version support
- Nexo item definitions
- Nexo pack structure and hosting

Revision #1
Created 2026-11-09 10:42:41 UTC by CloudExa Hosting Admin
Updated 2026-11-09 10:42:41 UTC by CloudExa Hosting Admin