Preserve Oraxen item IDs while changing a custom item texture

Change the appearance of an established Oraxen item as a controlled content update. Keep its internal identity separate from the name players see. Renaming an internal item key or replacing data files during a texture edit can affect integrations and items that already exist in inventories.

Identify the item and supported release

This guide concerns the Oraxen Bukkit plugin's item configuration. Check the installed release's Minecraft and Java requirements and use documentation for that release. Modern components and model definitions differ from older custom-model-data examples. Avoid copying another custom-item plugin's YAML because similar features do not imply the same schema.

Find the item under `plugins/Oraxen/items/`, where Oraxen stores YAML definitions. Record its internal configuration key, base material, model reference, display name and relevant mechanics. Search your own shop, recipe and quest configurations for that internal key. Keep those references unchanged during a visual-only edit.

Save a recoverable content set

1. Stop the server for a consistent backup. Preserve Oraxen configuration, item definitions, pack source assets and generated output, plus persistent mapping data used by your release.
2. Keep a copy of the world and player data containing the established item. A configuration-only backup cannot recover a changed inventory.
3. Copy the current item to a named test chest on a private server copy. Record its expected properties without exposing private NBT in public screenshots.

4. Keep the old pack download available privately for rollback. Avoid overwriting your only previous generated archive.

Edit the appearance, not the identity

Change one texture or its documented model reference in the item's pack configuration. Preserve the existing internal item key and intended base material. Oraxen distinguishes item properties, modern components, pack settings and mechanics; use the section responsible for appearance rather than changing an unrelated behaviour field.

Check filename case and namespace paths when uploading assets. A path that appears correct in a local case-insensitive file browser can differ from the filename on the server. Inspect the JSON model's texture references as well as the PNG itself. Keep copies of edited assets outside the pack build input so an accidental duplicate cannot override the intended file.

Compare existing and fresh copies

Use the documented reload and pack-generation procedure for your exact Oraxen release on the test server. Read generation output before distributing the result. Join with a matching Java client that accepts the server pack. Inspect the old chest item, then obtain a fresh copy using the release's administrative item interface.

Compare appearance, stack behaviour, name and intended mechanics. If fresh items work while old copies differ, investigate the release's item-update mechanism and migration notes before touching player inventories. Do not replace every old item as a shortcut; players may have legitimate durability, enchantment or naming changes.

Release or roll back

Publish the tested pack and matching configuration together during your planned change window. Keep the previous set until players have checked the result. If the change fails, restore the saved configuration and previous pack delivery settings. A failed texture update should not require deleting player items, changing their IDs or reinstalling the server.

Related CloudExa guides

back up the server files · check Java resource-pack delivery · plan a compatible Paper and plugin update

Official references

- Oraxen item configuration
- Oraxen source and release links

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