

FS25_ObjectStorageVisuals - Documentation

Quick guide for integrating the mod into a GIANTS-objectStorage.

Version 1.0.0.0

1. Requirements

FS25_ObjectStorageVisuals (OSV) extends GIANTS-objectStorage with additional visible representations and optional modder functions.

The functions are intentionally separated by storage type.

Storage type	XML detection	Extension block
Bale-only storage	supportsBales="true" and supportsPallets="false"	baleStorageVisuals
Pallet-only storage	supportsBales="false" and supportsPallets="true"	palletStacking
Mixed storage	supportsBales="true" and supportsPallets="true"	none

Bale-only storages are prepared using baleStorageVisuals:

```
<objectStorage supportsBales="true" supportsPallets="false" capacity="110">
  ...
  <baleStorageVisuals enabled="true">
    ...
  </baleStorageVisuals>
</objectStorage>
```

Pallet-only storages are prepared using palletStacking when additional settings are required:

```
<objectStorage supportsBales="false" supportsPallets="true" capacity="250">
  ...
  <palletStacking dynamicCapacity="true" hardLimit="500" />
</objectStorage>
```

Mixed storages are not prepared for these extensions. They remain with the normal GIANTS-objectStorage behavior.

The alignment of storage positions should generally be handled through the storageArea nodes in the i3d.

2. baleStorageVisuals

Entry inside objectStorage.

```
<baleStorageVisuals enabled="true" dynamicCapacity="true" hardLimit="500">
  ...
</baleStorageVisuals>
```

Parameter	Default	Function
enabled	false	Enables bale visuals for this storage.
dynamicCapacity	false	Checks the actually visible space in the storageAreas when storing objects.
hardLimit	500	Technical upper limit when dynamicCapacity is active.

baleStorageVisuals is intended for bale-only storages.

2.1 storageArea groups for bales

Bale-only storages can optionally assign their storageAreas to bale groups. This is done by adding the baleGroups attribute to a storageArea.

```
<objectStorage supportsBales="true" supportsPallets="false" capacity="650">
  <storageAreas>
    <storageArea startNode="storageAreaStartHay" endNode="storageAreaEndHay"
maxHeight="4.6" baleGroups="hay"/>
    <storageArea startNode="storageAreaStartStraw" endNode="storageAreaEndStraw"
maxHeight="4.6" baleGroups="straw"/>
    <storageArea startNode="storageAreaStartWrapped" endNode="storageAreaEndWrapped"
maxHeight="4.6" baleGroups="wrapped"/>
    <storageArea startNode="storageAreaStartFallback" endNode="storageAreaEndFallback"
maxHeight="4.6"/>
  </storageAreas>
</objectStorage>
```

Group	Meaning
hay	dry hay / dry forage bales
straw	straw bales
wrapped	wrapped bales

Multiple groups per storageArea are possible:

```
<storageArea startNode="..." endNode="..." maxHeight="4.6" baleGroups="hay straw"/>
```

A storageArea without baleGroups acts as a remaining or fallback area. It can accept bales that cannot be assigned to a matching group area.

Wrapped bales take priority over the fill type group. A wrapped grass, hay, clover or alfalfa bale therefore belongs to the wrapped group.

Unwrapped grass is not assigned to any of the groups hay, straw or wrapped. Such bales require an ungrouped fallback storageArea if they should be accepted.

Grouping is optional. Without baleGroups, the previous behavior remains unchanged.

Missing map- or mod-dependent fill types are ignored. This keeps the function map-safe and does not create additional log warnings.

When dynamicCapacity is active, the grouped areas are also used for the acceptance check. A hay bale is therefore not simply stored in a straw-only area just because there is still space there. If a matching group area is full, only an ungrouped fallback area can take over. If no suitable area is available, storing is refused.

3. slotTrigger for bale-only storages

For bale-only storages, there are two variants: a single slotTrigger and multiple group-specific slotTrigger entries via the slotTriggers block.

For grouped bale storages, the slotTriggers block is recommended.

3.1 Single slotTrigger

```
<slotTrigger enabled="true" movePlayerTrigger="true" playerTriggerOffset="0 0 0" />
```

Parameter	Default	Function
enabled	false	Enables the moving SlotTrigger.
movePlayerTrigger	false	Moves the PlayerTrigger together with the SlotTrigger.
playerTriggerOffset	0 0 0	Offset of the PlayerTrigger relative to the SlotTrigger, in meters.

The correct playerTriggerOffset depends on the individual storage and i3d structure.

When slotTrigger is active, the normal objectTrigger is replaced and hidden. Visible markers or warning stripes should therefore not be children of the old objectTrigger if they should remain visible.

The single slotTrigger is intended for classic storages with one moving storage point. movePlayerTrigger and playerTriggerOffset apply only to this single block.

Storage state	Trigger position
Storage empty	Start of the first storageArea
Storage partly filled	Next free storage position
Storage completely filled	Last occupied storage position

3.2 Group-specific slotTriggers

```
<baleStorageVisuals enabled="true" dynamicCapacity="true" hardLimit="650">
  <slotTriggers>
    <slotTrigger baleGroups="hay" />
    <slotTrigger baleGroups="straw" />
    <slotTrigger baleGroups="wrapped" />
  </slotTriggers>
</baleStorageVisuals>
```

Parameter	Default	Function
enabled	true	Enables or disables a single group SlotTrigger.
baleGroups	-	Restricts the trigger to one or more bale groups.

An entry with baleGroups="hay" accepts hay bales. An entry with baleGroups="straw" accepts straw bales. An entry with baleGroups="wrapped" accepts wrapped bales.

An entry without baleGroups acts as a fallback or remaining trigger, for example for unwrapped grass or unknown bale groups.

With enabled="false", an individual trigger can be disabled deliberately.

The trigger position is calculated from the existing storageAreas. No separate trigger nodes have to be specified in the XML.

Grouped triggers move to the next matching position of their group. If the matching group area is full and an ungrouped fallback storageArea exists, the grouped trigger may use that fallback area.

With multiple group-specific slotTriggers, there are not multiple GIANTS-playerTriggers. The GIANTS-playerTrigger remains the central interaction point for the storage menu.

4. palletUnderlays

Optional entry inside baleStorageVisuals.

```
<palletUnderlays enabled="true"
  positionVariation="0.025"
  rotationVariation="1.5"
  rotationY="0" />
```

Parameter	Default	Function
enabled	false	Shows underlay pallets below occupied bale areas.
positionVariation	0.025	Slight position variation of the underlay pallets.
rotationVariation	1.5	Slight rotation variation of the underlay pallets, in degrees.
rotationY	0	Fixed Y rotation of the underlay pallets, for example 90.

When `palletUnderlays` is used together with `dynamicCapacity`, OSV additionally checks whether a suitable underlay pallet area exists for newly occupied floor spaces. This prevents bales from being accepted beyond the last visible underlay pallet.

5. palletStacks

Entry inside `baleStorageVisuals`.

```
<palletStacks>
  <palletStack linkNode="stackRef01"
    translation="0 0 0"
    rotation="0 0 0"
    maxPallets="20"
    collision="true" />
</palletStacks>
```

Parameter	Default	Function
linkNode	-	Required. Reference node for the respective stack.
translation	0 0 0	Position relative to the linkNode.
rotation	0 0 0	Rotation relative to the linkNode, in degrees.
maxPallets	auto	Maximum pallet count for this stack.
collision	false	Enables collision for this stack.

6. Pallet storages

The pallet functions are intended for pallet-only storages.

Visible pallet stacking is enabled by default for pallet-only storages, as long as the respective pallet is prepared for stacking. `enabled="true"` therefore does not have to be set.

Example with dynamic pallet capacity:

```
<objectStorage supportsBales="false" supportsPallets="true" capacity="250">
  ...
  <palletStacking dynamicCapacity="true" hardLimit="500" />
</objectStorage>
```

Example without visible pallet stacking, but with a usable `SlotTrigger`:

```
<objectStorage supportsBales="false" supportsPallets="true" capacity="250">
  ...
  <palletStacking enabled="false" dynamicCapacity="true" hardLimit="500">
    <slotTrigger enabled="true" movePlayerTrigger="true" playerTriggerOffset="0 0 0" />
  </palletStacking>
</objectStorage>
```

Parameter	Default	Function
enabled	true	With false, only visible pallet stacking is disabled for this storage. dynamicCapacity and slotTrigger remain usable separately.
dynamicCapacity	false	Checks during storing whether there is still space in the visible pallet layout.
hardLimit	500	Technical upper limit when dynamic pallet capacity is active.
stackInMultiLevelStorage	false	Allows visible pallet stacking inside racks or multi-level storages with multiple storageAreas at different heights.
rotatePallets	false	Global switch for pallet rotation.
rotateEuroPallets	false	Rotates detected Euro pallets by 90 degrees.
rotateBigBagPallets	false	Rotates detected BigBag pallets by 90 degrees.
rotateSquarePallets	false	Rotates detected square pallets by 90 degrees, for example IBC / tank pallets.
rotateLargePallets	false	Rotates detected large or long pallets by 90 degrees.
rotateBigBags	false	Rotates pure BigBags by 90 degrees.

The dynamic pallet capacity only replaces the simple item count limit of the GIANTS-objectStorage. Fill types, farm permissions and all other GIANTS checks remain unchanged.

Mixed storages with bales and pallets are not prepared through palletStacking. They remain with the normal GIANTS-objectStorage behavior.

6.1 Pallet rotation

OSV can optionally rotate pallets by 90 degrees for the visible representation. Rotation is a modder option for the visible layout and rack / area adjustment.

It does not replace free storage rotation and does not automatically make unsuitable storageAreas fit.

```
<palletStacking
  dynamicCapacity="true"
  hardLimit="60"
  rotatePallets="true"
  rotateBigBagPallets="false" />
```

Parameter	Default	Function
rotatePallets	false	Global switch for pallet rotation.
rotateEuroPallets	false	Rotates detected Euro pallets.
rotateBigBagPallets	false	Rotates detected BigBag pallets.
rotateSquarePallets	false	Rotates detected square pallets, for example IBC / tank pallets.
rotateLargePallets	false	Rotates detected large or long pallets.
rotateBigBags	false	Rotates pure BigBags.

If rotatePallets is missing or false, only groups whose own group switch is explicitly set to true are rotated.

If rotatePallets="true" is set, all supported pallet groups are generally rotated. Individual groups can then be excluded again through their own group switch, if that switch is explicitly set to false.

```
<palletStacking
  rotatePallets="true"
  rotateBigBagPallets="false"
  rotateSquarePallets="true" />
```

In this example, pallets are generally rotated. BigBag pallets are explicitly excluded. Square pallets remain explicitly allowed.

The rotation must always match the actually defined storageArea. OSV does not artificially shrink object dimensions. If a rotated pallet does not fully fit into the storageArea, the corresponding rotation should be disabled or the storageArea should be adjusted.

6.2 Racks and multi-level storages

If a storage uses multiple storageAreas at different Y heights, OSV treats the storage as a multi-level storage. Visible pallet stacking is disabled by default in such storages, so pallets are not accidentally stacked additionally inside individual rack compartments.

The current threshold for automatic multi-level detection is a 0.35 m height difference between the storageArea start nodes.

If a pallet-only storage should deliberately be used as a rack with visible rack logic, this can be enabled:

```
<palletStacking
  dynamicCapacity="true"
  hardLimit="60"
  stackInMultiLevelStorage="true" />
```

With `stackInMultiLevelStorage="true"`, OSV treats each storageArea as its own rack compartment or rack segment.

DynamicCapacity and visual rebuild use the same rack-slot logic. This prevents the storage from reporting "full" even though visually useful rack positions are still available, or pallets being stored invisibly into unusable areas.

Identical pallets are preferably placed onto existing matching stacks or slots first. After that, free matching rack positions are used. This is especially important for mixed racks with IBCs, box pallets, consumable pallets and BigBagPallets.

6.3 Example for a pallet rack

```
<objectStorage supportsBales="false" supportsPallets="true" capacity="60">
  <storageAreas>
    <storageArea startNode="shelf01Start" endNode="shelf01End" maxHeight="1.65" />
    <storageArea startNode="shelf02Start" endNode="shelf02End" maxHeight="1.65" />
    <storageArea startNode="shelf03Start" endNode="shelf03End" maxHeight="1.65" />
  </storageAreas>
  <palletStacking
    dynamicCapacity="true"
    hardLimit="60"
    stackInMultiLevelStorage="true"
    rotateEuroPallets="true"
    rotateBigBagPallets="false"
    rotateSquarePallets="true" />
</objectStorage>
```

In this example, visible stacking inside the rack compartments is allowed. Euro pallets and square pallets may be rotated. BigBag pallets remain unrotated because in many racks they may become too deep or too wide for the compartment when rotated.

Modder notes for racks:

- Create storageAreas cleanly per compartment or level.
- Do not let storageAreas overlap unnecessarily.

- Set `maxHeight` per level realistically to the compartment height.
- Align the X/Z direction of the areas to the intended rack depth and rack width.
- Do not let storageAreas cut through posts, walls, rack floors or other collisions.
- `maxWidth` and `maxLength` on the `objectStorage` can still be used to exclude unsuitable pallets based on their Basegame object dimensions.

The size check through `maxWidth` and `maxLength` happens before the visual rotation and before the rack-slot visualization.

IBCs are approximately 1.35 x 1.35 in the Basegame. BigBagPallets are approximately 1.50 x 1.20.

```
<objectStorage maxWidth="1.40" maxLength="1.40">
```

A rack with this filter allows IBCs, but excludes BigBagPallets through the basic size check.

6.4 storageArea#maxHeight in pallet storages

Vanilla uses `storageArea#maxHeight` only in a limited way for pallet storages, because pallets are not visibly stacked in the normal GIANTS-objectStorage.

OSV uses this value as the visible stacking limit when `palletStacking` is active. In racks with `stackInMultiLevelStorage="true"`, `maxHeight` describes the maximum stack height inside the respective rack compartment.

The value should therefore be deliberately matched to the real compartment height.

```
<storageArea startNode="shelf01Start" endNode="shelf01End" maxHeight="1.65" />
```

If a rack compartment should only accept one layer, `maxHeight` must be set accordingly low. If two low pallet layers should be possible, `maxHeight` must be set accordingly higher.

6.5 Acceptance filter, visual footprint and stack height

OSV distinguishes between three things: the acceptance filter of the `objectStorage`, the visual footprint for visible placement and the stack height or allowed stack layers.

The basic acceptance check still uses the Basegame / Store size of the pallet. This includes, for example, `objectStorage#maxWidth`, `objectStorage#maxLength` and `objectStorage#maxHeight`.

The OSV stack data affects the visible representation, the used footprint in the visual rebuild, the stack height, the maximum number of layers and the rack-slot occupation.

A pallet can be excluded by the normal size check of the `objectStorage`, even if its visual stack data is smaller or set differently.

For racks, it can make sense to deliberately exclude certain pallet types via `maxWidth` or `maxLength` if they do not visually fit the rack.

7. slotTrigger for pallet-only storages

Optional entry inside `palletStacking`.

```
<objectStorage supportsBales="false" supportsPallets="true" capacity="250">
  ...
  <palletStacking dynamicCapacity="true" hardLimit="500">
    <slotTrigger enabled="true" movePlayerTrigger="true" playerTriggerOffset="0 0 0" />
  </palletStacking>
</objectStorage>
```

Parameter	Default	Function
enabled	false	Enables the moving SlotTrigger for pallet-only storages.
movePlayerTrigger	false	Moves the PlayerTrigger together with the SlotTrigger.
playerTriggerOffset	0 0 0	Offset of the PlayerTrigger relative to the SlotTrigger, in meters.

The pallet SlotTrigger uses the visible pallet placement of the storage.

Storage state	Trigger position
Storage empty	Start of the first storageArea
Storage partly filled	Next free pallet position
Storage completely filled	Last occupied pallet position

The correct playerTriggerOffset depends on the individual storage and i3d structure.

The pallet slotTrigger can also be used if visible pallet stacking has been disabled for a storage, for example in racks with their own storageArea levels.

For storages whose storage area should be walkable, blocking collisions must not be placed in the storage area.

8. objectStorageStacking for pallets from external mods

This block belongs in the XML of the respective pallet from an external mod. It is not entered in the objectStorage of the storage.

Simple example in the pallet XML:

```
<objectStorageStacking enabled="true" />
```

Extended example in the pallet XML:

```
<objectStorageStacking
  enabled="true"
  stackHeight="0.82"
  stackWidth="1.24"
  stackLength="0.84"
  maxStackLayers="2"
  stackFullOnly="true"
  fullFillLevel="1000" />
```

Parameter	Default	Function
enabled	false	Allows visible stacking of this pallet in the ObjectStorage.
stackHeight	StoreData	Vertical stack distance in meters.
stackWidth	StoreData	Width of the used visual footprint in meters.
stackLength	StoreData	Length of the used visual footprint in meters.
maxStackLayers	unlimited	Maximum visible stack layers.
stackFullOnly	false	Stacks only full pallets.
fullFillLevel	automatic	Reference value for full pallets.
palletGroup	automatic	Overrides the pallet group for rotation and footprint assignment.

stackHeight is a meter value. maxStackLayers is the number of allowed layers.

stackFullOnly="false" allows stacks of partially filled pallets with the same fill level. Depending on the model, this can cause visual collisions.

palletGroup only affects the group assignment for rotation and footprint logic. The entry does not automatically enable visible stacking.

Allowed palletGroup values:

```
euroPallet  
bigBagPallet  
squarePallet  
largePallet  
bigBag
```

Example for pure group assignment without enabled stacking:

```
<objectStorageStacking palletGroup="squarePallet" />
```

If a pallet should also be visibly stackable, enabled="true" must be set and the pallet must be prepared accordingly:

```
<objectStorageStacking  
  enabled="true"  
  palletGroup="euroPallet"  
  stackHeight="0.82"  
  maxStackLayers="2" />
```

This entry is intended for modders who want to prepare their own pallets from their mod for visible stacking in the ObjectStorage.

Only real GIANTS-objectStorage pallets are supported. No decoration objects, placeables or special objects.

9. Consumable pallets and variants

OSV corrects the client display of certain Basegame consumable pallets in the ObjectStorage.

Affected Basegame pallet	Description
baleTwinePallet	Bale twine pallet
baleNetPallet	Round bale net pallet
raniWrapPallet	Bale wrap pallet

In multiplayer, bale twine, bale net and bale wrap are displayed visibly again in the ObjectStorage on clients, instead of appearing only as an empty pallet.

Configured consumable pallets are no longer merged only by their base pallet. Variants, brands or colors are preserved for the visible representation, as far as they can be resolved from the pallet / consumable configuration.

The representation remains correct after storing, savegame reload and multiplayer synchronization.

The unloading dialog distinguishes configured pallet variants better when a real variant text can be resolved. With multiple variants of the same pallet, this prevents only identical generic entries from being shown and instead adds recognizable variant / brand information.

9.1 FS25_moreWrapColors

OSV supports FS25_moreWrapColors automatically.

MoreWrapColors bale wrap pallets are displayed correctly in the ObjectStorage. The additional colors work in Singleplayer, Selfhost / Host-MP and on Dedicated Server clients.

Full MoreWrapColors bale wrap pallets, half MoreWrapColors bale wrap pallets and additional bale wrap colors from the consumable configuration are taken into account.

10. Compatibility / Crossplay

The OSV entries are optional extensions.

Without the OSV script, the normal GIANTS-objectStorage remains usable. Only the additional visual, SlotTrigger and capacity functions are missing.

Objects already stored are not changed, deleted or migrated by OSV. OSV affects the acceptance of new objects and the visible representation, not the saved inventory.

Note about existing savegames

dynamicCapacity does not modify objects that are already stored.

If dynamicCapacity is enabled later or if storageAreas, hardLimit values or visible storage areas change, existing objects remain in the storage inventory. As a result, a storage can temporarily contain more objects than could be newly stored according to the new visible capacity check.

Unloading remains possible. New objects can only be stored again once the dynamic capacity check detects free visible storage space.

The OSV XML blocks do not change crossplay capability by themselves. Scripts and assets of the respective storage are what matters.

11. FAQ / Typical setup mistakes

Why can I no longer store anything even though the normal capacity has not been reached?

When dynamicCapacity is active, not only the technical objectStorage#capacity counts. OSV additionally checks whether visible space is still available in the defined storageAreas for the next object. If the visible area is full, storing is blocked even if the technical capacity is higher.

What is the difference between capacity, dynamicCapacity and hardLimit?

capacity is the normal GIANTS item count limit of the objectStorage. dynamicCapacity is the additional check of visible space in the storageAreas. hardLimit is an additional technical upper limit for storages with active dynamicCapacity.

Why does a storage remain overfilled after an update?

Already stored objects are not deleted or reduced. If a storage contains more objects after an update than the new dynamic check would allow, that inventory remains. Unloading still works. New storing is only possible again once enough visible storage space is free.

Why do pallets not visibly stack in my rack?

Visible pallet stacking is disabled by default in multi-level storages. This prevents racks or multi-level storages from accidentally stacking additionally inside every compartment. If stacking in the rack is intended, stackInMultiLevelStorage="true" must be set.

Why is a pallet not stored even though rotation is enabled?

The rotated pallet must fully fit into the defined storageArea with its real object dimensions. OSV does not artificially shrink pallets. If a pallet becomes too wide or too deep for a compartment when rotated, it is not considered a valid fit.

Why do pallets stand inside posts, walls or rack parts?

OSV only knows the defined storageAreas, not the full geometry of the building. The storageAreas must be set so that they do not cut through posts, walls, rack floors or other collisions. In racks, each compartment should have its own precisely limited storageArea.

Why is another pallet placed into the same compartment next to a stacked pallet?

When `stackInMultiLevelStorage="true"` is active, OSV continues to use the remaining free space inside a rack compartment as long as the next object fully fits into the `storageArea`. The compartment is not reserved globally for one object group.

Why do some pallet types stand next to each other and others do not?

OSV uses the real object dimensions and the defined `storageArea`. Different pallet types can have different width, length and height. What fits for Euro pallets does not automatically have to fit for BigBag pallets or IBC pallets.

Why are multiple bale wrap colors shown separately?

OSV takes pallet configurations into account for internal grouping and visual representation. This preserves variants, colors or brands as far as they can be resolved from the configuration.

Why does a pallet not fit into the rack despite stack data?

The basic acceptance check still uses the Basegame / Store dimensions of the object. OSV stack data controls visible placement, but does not replace the normal `maxWidth` / `maxLength` checks of the `objectStorage`.

Can I use these extensions for mixed storages with bales and pallets?

No. The OSV extensions `baleStorageVisuals` and `palletStacking` are intended for bale-only or pallet-only storages. Mixed storages remain with the normal GIANTS-objectStorage behavior.

What should I include in my own mod description as a modder?

If a storage uses `dynamicCapacity`, the mod description should briefly mention that the visible storage area limits the storage in addition to the normal item count capacity. Existing inventories remain intact. New storing can be blocked until visible storage space is free again.

12. Text note for modder mod descriptions

This text can be used in the description of a storage mod if the storage uses `dynamicCapacity`:

This storage uses a dynamic capacity check. Objects already stored in existing savegames remain intact. If the storage is technically or visually overfilled after an update, objects can still be unloaded. New objects can only be stored again once enough visible storage space is available.