

GetOffMyLawn Claim Mod - Complete Updated Development Guide

Project Overview

Goal: Recreate the "Claim" mod functionality for Minecraft 1.21.5 + NeoForge 21.5.81+, based on the original 1.21.1 version.

Project Name: GetOffMyLawn









Mod ID: `getoffmylawn`

Package: `com.mysparkle1991.getoffmylawn`








Architecture: Modern NeoForge with ModDevGradle + GUI System

Current Project Status

Core Infrastructure COMPLETE








-  Main mod class with proper registrations
-  Block system (BedrockClaimBlock + BlockEntity)
-  Item registration and creative tab
-  Player data system using Attachments (modern alternative to capabilities)
-  Game rules system (maxClaimCount, claimPermission)
-  Command structure with all major commands registered
-  Basic resource files (blockstates, models, lang)
-  **NEW:** Complete GUI system with multiple access methods

Working Features FULLY FUNCTIONAL




-  `/claim` - Basic chunk claiming (places bedrock block at Y=-63)
-  `/claim info` - Shows detailed claim information with owner, date, category
-  `/claimgui` - Gives player the GUI item for claim management
-  **GUI System** - Visual interface for all claim operations
 - Right-click GUI item to open anywhere
 - Right-click claim blocks to open directly
 - Context-aware interface (different buttons for owners vs visitors)
 - Real-time claim status detection
-  Player claim count tracking with game rule limits
-  Permission system (respects game rules and OP status)
-  Claim data storage in block entity NBT

-  Proper chunk coordinate detection

Stub Commands REGISTERED BUT NOT IMPLEMENTED

-  `/claim admin add/remove/list` - Admin management system
-  `/claim delete` - Claim deletion (GUI button exists but uses stub)
-  `/claim transfer <player>` - Ownership transfer
-  `/claim rename <name>` - Claim renaming (GUI inputs exist but not functional)
-  `/claim setcategory <category>` - Category system
-  `/claim reset` - Reset to defaults
-  `/claim type public/private` - Access control (GUI buttons exist but use stubs)

Critical Missing HIGH PRIORITY

-  **Protection Systems** - Players can still break/place blocks in claims
-  **Admin System Implementation** - Full admin management
-  **Visual Claim Names** - Show claim names when entering chunks

Complete File Structure Status

src/main/java/com/mysparkle1991/getoffmylawn/

- |— GetOffMyLawn.java  COMPLETE - Main mod class with GUI integration
- |— block/
 - |— BedrockClaimBlock.java  COMPLETE - Right-click opens GUI
- |— block/entity/
 - |— BedrockClaimBlockEntity.java  COMPLETE - Data storage
- |— client/
 - |— ClientEvents.java  COMPLETE - GUI screen registration
- |— command/
 - |— ClaimCommand.java  COMPLETE - All commands registered
 - |— ClaimGUICommand.java  COMPLETE - /claimgui command
- |— gui/
 - |— ClaimMenu.java  COMPLETE - Server-side container
 - |— ClaimScreen.java  COMPLETE - Client-side visual interface
- |— init/
 - |— ClaimModBlocks.java  COMPLETE
 - |— ClaimModBlockEntities.java  COMPLETE
 - |— ClaimModItems.java  COMPLETE - Includes GUI item
 - |— ClaimModMenuTypes.java  COMPLETE - GUI registration
 - |— ClaimModGameRules.java  COMPLETE
- |— item/
 - |— ClaimGUIItem.java  COMPLETE - Craftable GUI access item
- |— network/
 - |— ClaimModVariables.java  COMPLETE - Player data attachments
- |— procedures/
 - |— CheckIfChunkXYZIsClaimedProcedure.java  COMPLETE
 - |— YCoordsOfBedrockClaimProcedure.java  COMPLETE
 - |— ClaimChunkProcedure.java  COMPLETE
 - |— ClaimChunkXYZForEntityProcedure.java  COMPLETE
 - |— SetClaimDataProcedure.java  COMPLETE
 - |— ClaimInfoProcedure.java  COMPLETE - Shows detailed info
 - |— ClaimAddAdminProcedure.java  STUB - "Admin system coming soon!"
 - |— ClaimRemoveAdminProcedure.java  STUB
 - |— ClaimAdminListProcedure.java  STUB
 - |— ClaimDeleteProcedure.java  STUB - "Delete function coming soon!"
 - |— ClaimTransferProcedure.java  STUB
 - |— ClaimRenameProcedure.java  STUB
 - |— ClaimSetCategoryProcedure.java  STUB
 - |— ClaimResetProcedure.java  STUB
 - |— SetPublicProcedure.java  STUB - "Public/Private system coming soon!"
 - |— SetPrivateProcedure.java  STUB

```

src/main/resources/
├── assets/getoffmylawn/
│   ├── lang/en_us.json ✓ COMPLETE - Includes GUI translations
│   ├── blockstates/bedrock_claim.json ✓ COMPLETE
│   └── models/
│       ├── block/bedrock_claim.json ✓ COMPLETE
│       └── item/
│           ├── bedrock_claim.json ✓ COMPLETE
│           └── claim_gui.json ✓ COMPLETE - Uses compass texture
└── data/getoffmylawn/
    ├── recipes/
    └── claim_gui.json ✓ COMPLETE - Stone+Glass+Compass recipe

```

Architecture & Design Decisions

Player Data System

- **Modern Approach:** Uses NeoForge Attachments instead of deprecated capabilities
- **Data Structure:**

```

java
PlayerVariables {
    playerClaimCount: double // Number of claims owned
    lastClaimName: String // Last entered claim name
    lastX/lastZ: double // Last chunk coordinates for name display
}

```

Claim Data Storage

- **Location:** Block entity NBT at bedrock claim blocks (Y=-63)
- **Data Structure:**

java

```
BlockEntity NBT {  
    "owneruuid": String    // Player UUID who owns claim  
    "ownerdisplay": String // Player display name  
    "claimname": String    // Custom claim name  
    "category": String     // Claim category  
    "Cdate": String       // Creation date (DD/MM/YYYY)  
    "admincount": Double   // Number of admins  
    "admin{i}": String     // Admin UUID (indexed 0,1,2...)   
    "isadmin{i}": Boolean  // Admin active status  
    "displayadmin{i}": String // Admin display name  
    "Cpublic": Boolean     // Public/private access  
}
```

GUI System Architecture

- **Three Access Methods:**
 1. Craftable item (Stone+Glass+Compass) → Right-click anywhere
 2. Command `/claimgui` → Gives GUI item
 3. Direct access → Right-click any claim block
- **Context-Aware Interface:** Different buttons based on ownership and claim status
- **Real-time Updates:** Reads current chunk data on GUI open
- **Integrated with Commands:** GUI buttons execute same procedures as commands

Chunk Detection System

- **Coordinate System:** Uses `LevelChunk.getPos().x/z` for chunk coordinates
- **Claim Block Placement:** Always at Y=-63 regardless of terrain
- **Detection Logic:** Checks for bedrock claim block at chunk corner coordinates

Priority Implementation Order

PHASE 1: Protection Systems ❌ CRITICAL PRIORITY

Status: Not implemented - players can break blocks in claimed areas

Files needed:

procedures/

- |— ClaimBlockBreakProtectProcedure.java
- |— ClaimBlockPutProtectProcedure.java
- |— ClaimRightclickProtectProcedure.java
- |— ClaimBonemealProtectProcedure.java
- |— ClaimMultiblockPutProtectProcedure.java
- |— CheckIfUUIDAdminProcedure.java (utility for permission checking)

Event handlers required:

- `BlockEvent.BreakEvent` - Prevent unauthorized block breaking
- `BlockEvent.EntityPlaceEvent` - Prevent unauthorized block placing
- `PlayerInteractEvent.RightClickBlock` - Prevent unauthorized interactions
- `BonemealEvent` - Prevent unauthorized bonemeal use
- `BlockEvent.EntityMultiPlaceEvent` - Prevent unauthorized multi-block placement

Logic: Check if player is owner, admin, or if claim is public before allowing actions

Registration: Must add event handler registrations to main mod class:

```
java
```

```
NeoForge.EVENT_BUS.register(ClaimBlockBreakProtectProcedure.class);
```

```
// etc for each protection class
```

PHASE 2: Admin System Implementation ● MEDIUM PRIORITY

Status: Commands and GUI buttons exist but show "coming soon" messages

Files to implement:

procedures/

- |— ClaimAddAdminProcedure.java - Replace stub with full implementation
- |— ClaimRemoveAdminProcedure.java - Replace stub
- |— ClaimAdminListProcedure.java - Replace stub
- |— FindAdminNumberProcedure.java - New utility for admin management

Required logic:

- Add/remove players to admin list in block entity NBT
- Update admin count and indexed admin entries
- Check admin permissions in protection systems
- Display admin lists with proper formatting

- Handle admin UUID → display name conversion
- Integrate with GUI admin management button

PHASE 3: Claim Management Features 🟡 MEDIUM PRIORITY

Status: Commands exist, GUI inputs/buttons exist, but use stub implementations

Files to implement:

procedures

- |— ClaimDeleteProcedure.java - Replace stub (GUI delete button uses this)
- |— ClaimRenameProcedure.java - Replace stub (GUI name input uses this)
- |— ClaimSetCategoryProcedure.java - Replace stub (GUI category input uses this)
- |— SetPublicProcedure.java - Replace stub (GUI public button uses this)
- |— SetPrivateProcedure.java - Replace stub (GUI private button uses this)
- |— ClaimTransferProcedure.java - Replace stub
- |— ClaimResetProcedure.java - Replace stub
- |— SetResetProcedure.java - New utility for reset functionality

PHASE 4: Visual Claim System 🟢 LOW PRIORITY

Status: Not implemented

Files needed:

procedures/

- |— ShowClaimNameProcedure.java - Display claim names on chunk entry

Required logic:

- Player tick event to detect chunk changes
- Display title with claim name when entering chunks
- Show "Wildlands" for unclaimed areas
- Track player position to prevent spam
- Update player variables for last position

Critical Code Patterns

Protection Check Pattern (Used throughout protection systems)

java

// Standard permission check - use this exact pattern

```
double cY = YCoordsOfBedrockClaimProcedure.execute(world, x, y, z);
if (cY != -999.0) { // Chunk is claimed
    double cX = (world.getChunk(new BlockPos((int)x, (int)y, (int)z)).getPos()).x;
    double cZ = (world.getChunk(new BlockPos((int)x, (int)y, (int)z)).getPos()).z;

    BlockPos claimPos = BlockPos.containing(cX, cY, cZ);
    BlockEntity blockEntity = world.getBlockEntity(claimPos);
    if (blockEntity != null) {
        String ownerUUID = blockEntity.getPersistentData().getString("owneruuid");
        boolean isPublic = blockEntity.getPersistentData().getBoolean("Cpublic");

        // Check permissions: owner, admin, or public claim
        if (!CheckIfUUIDAdminProcedure.execute(world, cX, cY, cZ, entity.getStringUUID()) &&
            !ownerUUID.equals(entity.getStringUUID()) &&
            !isPublic) {
            // Deny action
            if (entity instanceof Player player) {
                player.displayClientMessage(Component.literal("$cYou cannot [action] here."), true);
            }
            if (event instanceof ICancellableEvent cancellable) {
                cancellable.setCanceled(true);
            }
        }
    }
}
```

Block Entity NBT Access Pattern

java

```
// Always use this pattern for NBT access
if (!world.isClientSide()) {
    BlockPos bp = BlockPos.containing(x, y, z);
    BlockEntity blockEntity = world.getBlockEntity(bp);
    BlockState bs = world.getBlockState(bp);
    if (blockEntity != null) {
        // Read
        String value = blockEntity.getPersistentData().getString("key");
        // Write
        blockEntity.getPersistentData().putString("key", "value");
    }
    // ALWAYS update block state after changes
    if (world instanceof Level level) {
        level.sendBlockUpdated(bp, bs, bs, 3);
    }
}
```

Event Handler Registration Pattern

```
java

// In main mod class constructor
NeoForge.EVENT_BUS.register(ProtectionProcedureClass.class);

// In procedure class
@EventBusSubscriber
public class ClaimBlockBreakProtectProcedure {
    @SubscribeEvent
    public static void onBlockBreak(BlockEvent.BreakEvent event) {
        execute(event, event.getLevel(), event.getPos().getX(), event.getPos().getY(), event.getPos().getZ(), event.getPlayer())
    }

    private static void execute(@Nullable Event event, LevelAccessor world, double x, double y, double z, Entity entity) {
        // Protection logic here
    }
}
```

Testing Checklist

Basic Functionality ☒ WORKING

- ☒ `/claim` claims a chunk and places bedrock block
- ☒ `/claim info` shows correct detailed claim information
- ☒ `/claimgui` gives GUI item

- ✓ GUI opens from item, command, and claim block interaction
- ✓ GUI shows correct claim status and owner information
- ✓ Game rules work: `/gamerule maxClaimCount X`
- ✓ Permission system works for OPs vs regular players
- ✓ Player claim count increases correctly
- ✓ Claim data persists (owner, name, date, category)

GUI System **WORKING**

- ✓ Can craft GUI item (Stone+Glass+Compass)
- ✓ Right-click GUI item opens interface
- ✓ Right-click claim blocks opens interface
- ✓ GUI shows different options for owners vs visitors
- ✓ GUI detects claim status in real-time
- ✓ Claim button works for unclaimed chunks
- ✓ Info button shows claim details

Stub Features **SHOWS "COMING SOON" MESSAGES**

- ✓ Admin management commands show placeholder messages
- ✓ Delete/transfer/rename commands show placeholder messages
- ✓ Public/private commands show placeholder messages
- ✓ GUI buttons for unimplemented features show placeholder messages

Protection Systems **TO BE TESTED AFTER IMPLEMENTATION**

- ☐ Cannot break blocks in claimed chunks (unless owner/admin/public)
- ☐ Cannot place blocks in claimed chunks (unless owner/admin/public)
- ☐ Cannot interact with blocks in claimed chunks (unless owner/admin/public)
- ☐ Public claims allow all interactions
- ☐ Admins have proper permissions in protection systems

Known Issues & Solutions

Issue: Protection not working

Status: Expected - protection systems not implemented yet **Solution:** Implement Phase 1 protection procedures with event handlers

Issue: GUI shows "coming soon" for some features

Status: Expected - stub implementations

Solution: Replace stub procedures with full implementations

Issue: Custom texture missing for GUI item

Status: Resolved - using compass texture temporarily **Fix Applied:** Changed model to use

minecraft:item/compass

Issue: Attachments not syncing

Solution: Ensure `syncPlayerVariables()` is called after data changes

Issue: Block entity data not persisting

Solution: Always call `level.sendBlockUpdated()` after NBT changes

Immediate Next Steps for Continuation

START HERE: Protection Systems (Critical Priority)

1. **Create `CheckIfUUIDAdminProcedure.java`** - Permission checking utility
2. **Implement `ClaimBlockBreakProtectProcedure.java`** - Prevent unauthorized breaking
3. **Add event handler registration** to main mod class
4. **Test protection** - try breaking blocks in claimed areas
5. **Expand to other protection types** (place, interact, bonemeal, multiblock)

Code Template for Protection System:

```
java

@EventBusSubscriber
public class ClaimBlockBreakProtectProcedure {
    @SubscribeEvent
    public static void onBlockBreak(BlockEvent.BreakEvent event) {
        execute(event, event.getLevel(), event.getPos().getX(), event.getPos().getY(), event.getPos().getZ(), event.getPlayer())
    }

    public static void execute(@Nullable Event event, LevelAccessor world, double x, double y, double z, Entity entity) {
        if (entity == null) return;







        // Use protection pattern from above
        // Check if chunk claimed, get permissions, cancel if unauthorized
    }
}
```

Build Commands

bash

```
./gradlew build      # Build the mod  
./gradlew runClient  # Test in development environment  
./gradlew runServer  # Test server functionality
```

Current State Summary

-  **Fully functional claiming system** with commands and GUI
-  **Complete GUI interface** with multiple access methods
-  **Modern NeoForge architecture** with proper registrations
-  **Player data tracking** and game rule integration
-  **No protection** - this is the critical missing piece
-  **Feature stubs** ready for implementation

The mod has a solid foundation and working GUI. The next critical step is implementing protection systems to prevent unauthorized actions in claimed areas.