# Developer API

**Repository Information**

Maven:

```
<repository>
    <id>mysticalkingdoms-public</id>
    <url>https://nexus.mysticalkingdoms.com/repository/public/</url>
</repository>
```

Gradle:

```
    maven {
        url = uri('https://nexus.mysticalkingdoms.com/repository/public/')
    }
```

To create your own expansion, you first need to have a main class that will handle enabling, disabling and reloading the expansion, just like a plugin. This is an example of a main class for an expansion:

```
package com.mysticalkingdoms.pouchexpansion.cratereloaded;

import com.mysticalkingdoms.missionpouches.MissionPouches;
import com.mysticalkingdoms.missionpouches.missions.MissionExpansion;
import com.mysticalkingdoms.missionpouches.missions.MissionExpansionDescription;
import com.mysticalkingdoms.pouchexpansion.cratereloaded.types.CrateReloadedOpen;

import java.io.File;

public class CrateReloadedExpansion extends MissionExpansion {

    public CrateReloadedExpansion(MissionPouches plugin, MissionExpansionDescription expansionDescription, 
File dataFolder) {
        super(plugin, expansionDescription, dataFolder);
    }

    @Override
    protected boolean onEnable() {
        registerMissionTypes(new CrateReloadedOpen(getPlugin()));
        return true;
    }

    @Override
    protected void onDisable() {}

    @Override
    protected void onReload() {}
}
```

Keep the constructor as is, otherwise your expansion will not load.

- **onEnable:** You should register any logic here, and then register your mission types.
- **onDisable:** Disable any logic that needs to be disabled (example: databases)
- **onReload:** This is called when /mp reload is used.

Now, for your mission type, you need to extend the class "MissionType".

There are two ways to create this class. The first one uses a Function that gives you a ConfigurationSection (the pouch configuration), and expects MissionProgress as the return value.

You must specify a mission type key. This should be unique. A good idea is to prefix it with the name of your plugin/expansion.

```
package com.mysticalkingdoms.pouchexpansion.cratereloaded.types;

import com.hazebyte.crate.api.event.CrateRewardEvent;
import com.mysticalkingdoms.missionpouches.MissionPouches;
import com.mysticalkingdoms.missionpouches.missions.MissionProgress;
import com.mysticalkingdoms.missionpouches.missions.MissionType;
import org.bukkit.event.EventHandler;

import java.util.List;

public class CrateReloadedOpen extends MissionType {
    public CrateReloadedOpen(MissionPouches plugin) {
        super(plugin, "CRATERELOADED_OPEN");
    }

    @EventHandler
    public void onReward(CrateRewardEvent event) {
        checkPouches(event.getPlayer(), section -> {
            if (section == null) {
                return MissionProgress.of(1L, true);
            }

            List<String> crates = section.getStringList("crates");
            if (crates.isEmpty()) {
                return MissionProgress.of(1L, true);
            }

            return MissionProgress.of(crates.contains(event.getCrate().getCrateName()) ? 1L : 0L, true);
        });
    }
}
```

MissionType implements Listener, so you can listen to any Bukkit events. In the example above, we're listening to CrateRewardEvent, and then calling checkPouches on the player, we get the ConfigurationSection for the pouch configuration, we do our checks, and then finally return MissionProgress. MissionProgress has 2 static methods:

- *of(long progress, boolean add):* This method sets the progress to the specified value. If the add boolean is set to true, then it will add progress instead of setting.
- *noProgress():* This method will just return no progress.

**Notes:**

1. The ConfiguratIonSection from the function can be null, for example, if the user didn't specify any parameters.

Now, there's a second version which gives you access to the NBTItem of the pouch when it's being checked.

This is an example of an expansion that uses it:

```
package com.mysticalkingdoms.pouchexpansion.uniquekills.types;

import de.tr7zw.changeme.nbtapi.NBTList;
import com.mysticalkingdoms.missionpouches.MissionPouches;
import com.mysticalkingdoms.missionpouches.missions.MissionProgress;
import com.mysticalkingdoms.missionpouches.missions.MissionType;
import org.bukkit.entity.Player;
import org.bukkit.event.EventHandler;
import org.bukkit.event.entity.PlayerDeathEvent;

import java.util.UUID;

public class UniqueKillsType extends MissionType {
    public UniqueKillsType(MissionPouches plugin) {
        super(plugin, "UNIQUEKILLS_KILL");
    }

    @EventHandler
    public void onKill(PlayerDeathEvent event) {
        Player killer = event.getEntity().getKiller();
        if (killer == null) {
            return;
        }

        checkPouches(killer, (section, nbtItem) -> {
            NBTList<UUID> list = nbtItem.getUUIDList("PlayersKilled");
            if (list.contains(event.getEntity().getUniqueId())) {
                return MissionProgress.noProgress();
            }

            list.add(event.getEntity().getUniqueId());
            return MissionProgress.of(1L, true);
        });
    }
}
```

This allows you to query the item for specific things, to make your expansions even more flexible. For example, the UniqueKills expansion, available on Discord, uses this to write NBT data of which players have been killed, so that you have to kill unique players to progress through the mission.

If you have any questions, feel free to ask on Discord!

Revision #1
Created 3 February 2024 14:58:58 by Admin
Updated 3 February 2024 15:15:59 by Admin