How to Diagnose Crash Reports

Crash reports are a critical tool in diagnosing issues with your Minecraft server. They provide insights into what caused a server or client crash, helping you resolve the issue quickly. In this guide, we will walk you through the process of understanding, interpreting, and using crash reports to troubleshoot and fix problems on your Minecraft server.

Accessing Crash Reports

Via Server Control Panel:

1. **Log into your server panel**: First, log into your Control Panel.
2. **Navigate to the Files section**: Once inside your server's dashboard, go to the **Files** section.
3. **Locate the crash reports folder**: Crash reports are stored in the **`crash-reports/`** directory of your server files. This folder contains text files named with the date and time of the crash (e.g., `crash-2025-01-01_16.00.00-server.txt`).

Via SFTP:

1. **Connect to Your Server via SFTP**:
   - Use an FTP client (e.g., FileZilla) to connect to your server.
   - Enter your FTP credentials provided by your hosting service.
2. **Navigate to the Crash Reports Folder**:
   - Locate the `crash-reports` folder in your server's root directory.
3. **Download the Latest Report**:
   - Download the latest crash report file to your computer for analysis.

Understanding Crash Reports

Each crash report provides detailed information about the crash event, and understanding its structure is key to diagnosing the problem. Here’s a breakdown of the common sections in a crash report:

Key Sections of a Crash Report:

- **Head Section**: This section includes basic server information, such as Minecraft version, server software (e.g., PaperMC, Spigot), Java version, and any environment details that might be relevant.
- **Stack Trace**: The stack trace is the most important part of the report. It shows the sequence of method calls that led to the crash. This is where you’ll find references to specific plugins, mods, or Minecraft’s own code that triggered the crash.
- **Cause Section**: This part will often mention the specific cause of the crash, such as a `NullPointerException`, a resource overload, or a mod/plugin conflict.

Example:

```
---- Minecraft Crash Report ----
// Why did you do that?

Time: 1/1/2025 4:00 PM
Description: Exception in server tick loop

java.lang.NullPointerException: Unexpected error
    at com.mojang.minecraft.server.MinecraftServer.run(MinecraftServer.java:500)
    at java.lang.Thread.run(Thread.java:748)
```

In this case:

- The **NullPointerException** is the type of error.
- The crash occurred in the **MinecraftServer.run** method.
- This indicates an issue in server execution, possibly with a mod or plugin.

Diagnosing Common Crash Causes

Mod or Plugin Issues

- **Symptoms**: The crash report references a specific mod or plugin, often with an error such as `NullPointerException`, `ClassNotFoundException`, or `NoClassDefFoundError`.
- **Steps to Diagnose**:
  1. Open the crash report and look for references to a mod or plugin in the stack trace.
  2. Ensure that all mods/plugins are compatible with your Minecraft version and up-to-date.
  3. If a recently added mod or plugin caused the crash, remove it and restart the server.

Example Error:

```
Caused by: java.lang.NullPointerException: null
    at com.example.mod.ModClass.someMethod(ModClass.java:42)
```

In this case, you may need to update or disable **ModClass**.

2. **World Corruption**

- **Symptoms**: Crashes related to world generation or chunk loading are often caused by corrupted world files.
- **Steps to Diagnose**:
  1. Open the crash report and check for keywords like “World Generation” or “Chunk Loading”.
  2. Backup the world folder and test with a clean, new world to see if the crash persists.
  3. Use tools like **WorldEdit** or **RegionFixer** to repair corrupted chunks.

Example Error:

```
at net.minecraft.world.gen.ChunkGenerator.func_225535_a_(ChunkGenerator.java:122)
```

This points to an issue during world generation or chunk loading, likely due to a corrupted world file.

3. **Out of Memory (OOM) Issues**

- **Symptoms**: Crashes caused by memory limitations will show a `java.lang.OutOfMemoryError` in the crash report.
- **Steps to Diagnose**:
  1. Look for the **OutOfMemoryError** in the crash report.
  2. Increase the amount of RAM allocated to your server in your **startup parameters**.
     - Edit your start script to allocate more memory, for example:

       ```
       java -Xmx4G -Xms2G -jar paper.jar
       ```

  3. If the issue persists, reduce the number of plugins or mods, optimize server settings, or upgrade the server.

Example Error:

```
java.lang.OutOfMemoryError: Java heap space
```

This indicates your server ran out of allocated memory and needs more resources to run properly.

4. **Plugin or Mod Conflicts**

- **Symptoms**: Crashes involving conflicting plugins or mods will show multiple references to conflicting classes or methods in the stack trace.
- **Steps to Diagnose**:
  1. Check the crash report for any repeated references to the same classes or methods across different plugins/mods.
  2. Disable or remove half of your installed plugins or mods to narrow down the source of the conflict.
  3. Update or replace conflicting plugins.

Example Error:

```
at org.bukkit.plugin.java.JavaPluginLoader.loadPlugin(JavaPluginLoader.java:78)
```

This points to a potential conflict during plugin loading. It may be necessary to update or disable one of the conflicting plugins.

Fixing the Issue Based on the Crash Report

1. **Research the Error**

- Search for the error message or stack trace online. Communities like **Spigot**, **PaperMC**, and **Minecraft forums** can provide insights into common crashes and solutions.

2. **Update Software**

- Ensure that your Minecraft version, server software (PaperMC, Spigot), Java, and any mods/plugins are up-to-date. Incompatibilities between versions are a common cause of crashes.

3. **Test in a Controlled Environment**

- If you suspect a plugin or mod, disable or remove them one at a time and restart the server. Testing in a controlled environment can help you isolate the issue.

4. **Check Server Logs**

- In addition to crash reports, check the server logs (**logs/latest.log**) for additional clues that may not be listed in the crash report.

Preventing Future Crashes

1. **Regular Backups**

- Always make regular backups of your server before installing new plugins, mods, or updates. This allows you to restore the server to a previous working state if a crash occurs.

2. **Test New Plugins or Mods**

- Before installing new plugins or mods, test them in a separate environment to ensure they don’t cause crashes on your live server.

3. **Optimize Server Performance**

- Use performance-monitoring plugins like **Spark** to keep track of performance and optimize your server where you can to prevent future resource-related crashes.