# Verbose Logging im WebSphere Liberty Server

Um Ihre Anfrage bezüglich der ausführlichen Protokolloption für den Server-Befehl im WebSphere Liberty Profile (WLP) zu beantworten, klären wir, was der `server`-Befehl tut und wie die Ausführlichkeit gesteuert werden kann.

Der `server`-Befehl in WLP wird verwendet, um Serverinstanzen zu verwalten, mit Aktionen wie `start`, `stop`, `run` oder `status`. Zum Beispiel:

- `server start <serverName>` startet den Server im Hintergrund, wobei die Ausgabe in Protokolldateien umgeleitet wird (z. B. `logs/console.log`).
- `server run <serverName>` startet den Server im Vordergrund und zeigt die Ausgabe direkt im Terminal an.

Ihre Frage bezieht sich auf eine "ausführliche Protokolloption"für den `server`-Befehl, was entweder eine detailliertere Ausgabe des Befehls selbst oder eine detailliertere Protokollierung des von ihm verwalteten Servers implizieren könnte. Nach der Untersuchung der Optionen ist klar, dass der `server`-Befehl kein direktes Flag wie `--verbose` oder `-v` hat, um seine eigene Ausführlichkeit zu erhöhen. Stattdessen bezieht sich die Ausführlichkeit auf das Protokollierungsverhalten des Servers, das beim Aufrufen des Befehls beeinflusst werden kann.

### Aktivierung ausführlicher Protokollierung

In WLP wird die Ausführlichkeit der Protokolle durch die Protokollierungskonfiguration des Servers und nicht direkt über ein Flag des `server`-Befehls gesteuert. Hier ist, wie Sie eine ausführliche Protokollierung aktivieren können:

**1. Protokollierung in `server.xml` konfigurieren** Die primäre Methode zur Aktivierung einer ausführlichen Protokollierung besteht darin, das `<logging>`-Element in der `server.xml`-Datei des Servers anzupassen, die sich normalerweise unter `<WLP_HOME>/usr/servers/<serverName>/` befindet. Sie können eine detaillierte Trace-Spezifikation festlegen, um die Protokollausführlichkeit zu erhöhen. Zum Beispiel:

```
<logging traceSpecification="*=all" />
```

- `*=all` aktiviert alle Trace-Punkte, wodurch die Protokolle extrem ausführlich werden (nützlich für das Debugging).
- Für eine gezieltere Ausführlichkeit können Sie Komponenten angeben, z. B. `*=info:com.example.*=debug`, wobei die Standardstufe auf `info` und `debug` für das `com.example`-Paket gesetzt wird.

Weitere nützliche Attribute sind: - `logLevel`: Setzt die allgemeine Protokollstufe (z. B. `INFO`, `DEBUG`, `TRACE`). - `consoleLogLevel`: Steuert die Stufe der Nachrichten, die in `console.log` oder dem Terminal geschrieben werden (z. B. `DEBUG` oder `TRACE`).

Beispiel mit einer feineren Konsolenstufe:

```
<logging consoleLogLevel="DEBUG" traceSpecification="*=audit" />
```

Wenn Sie `server start` ausführen, gehen die Protokolle (einschließlich der ausführlichen Ausgabe) in `logs/console.log`. Mit `server run` erscheint diese ausführliche Ausgabe direkt in Ihrem Terminal.

**2. Umgebungsvariablen verwenden** Sie können die Protokollausführlichkeit auch über Umgebungsvariablen steuern, die die Einstellungen in `server.xml` überschreiben oder ergänzen. Legen Sie beispielsweise die Variable `WLP_LOGGING_CONSOLE_LOGLEVEL` vor dem Ausführen des `server`-Befehls fest:

```
export WLP_LOGGING_CONSOLE_LOGLEVEL=DEBUG
server start <serverName>
```

- Dies setzt die Konsolenprotokollstufe auf `DEBUG`, wodurch eine detailliertere Ausgabe gewährleistet wird.
- Gültige Stufen sind `INFO`, `DEBUG`, `FINE`, `FINER`, `FINEST` usw.

Für `server run` wird die Terminalausgabe ausführlicher; für `server start` beeinflusst dies `console.log`.

**3. JVM-spezifische ausführliche Optionen** Wenn Sie an spezifischen ausführlichen Ausgaben wie Garbage-Collection-Protokollen interessiert sind, können Sie JVM-Optionen hinzufügen (obwohl dies serverbezogen ist und nicht der `server`-Befehl selbst). Fügen Sie `-verbose:gc` zur `jvm.options`-Datei in `<WLP_HOME>/usr/servers/<serverName>/` hinzu oder schließen Sie es über `server.xml` ein:

```
<jvmEntries verbose="gc" />
```

Dies erzeugt GC-bezogene Protokolle, die normalerweise in `console.log` oder dem Terminal mit `server run` erscheinen.

**Wie dies mit dem `server`-Befehl zusammenhängt**

- **Keine direkte `--verbose`-Option**: Der `server`-Befehl selbst (z. B. `server start`) hat kein `--verbose`- oder ähnliches Flag, um seine Ausführungsschritte über grundlegende Statusmeldungen (z. B. “Server gestartet”) hinaus zu detaillieren.
- **Server-Protokollierung steuert die Ausführlichkeit**: Die Ausführlichkeit, die Sie sehen, hängt von der Protokollierungskonfiguration des Servers ab. Verwenden Sie `server run`, um Protokolle live im Terminal zu beobachten, oder überprüfen Sie Protokolldateien nach `server start`.

Zum Beispiel: - Mit `<logging traceSpecification="*=all"/>` in `server.xml` flutet `server run <serverName>` das Terminal mit detaillierten Protokollen. - Mit `server start <serverName>` erscheint dieselbe Ausführlichkeit in `logs/console.log`.

**Empfehlung**

Um eine ausführliche Protokollierung beim Verwenden des `server`-Befehls zu erreichen: 1. Bearbeiten Sie `server.xml`, um eine detaillierte `traceSpecification` (z. B. `*=all` für maximale Ausführlichkeit oder eine spezifische Stufe wie `*=debug`) einzuschließen. 2. Alternativ setzen Sie `WLP_LOGGING_CONSOLE_LOGLEVEL=DEBUG` in Ihrer Umgebung. 3. Verwenden Sie `server run`, um die ausführliche Ausgabe direkt zu sehen, oder `server start` und überprüfen Sie `logs/console.log`.

Dieser Ansatz stellt sicher, dass Sie die detaillierten Protokolle erhalten, die Sie benötigen, und sie auf Ihre Debugging- oder Überwachungsanforderungen abgestimmt sind.

Wenn Sie etwas Spezielleres meinten (z. B. Ausführlichkeit der Befehlsausführung anstelle von Serverprotokollen), lassen Sie es mich bitte wissen! Andernfalls ist die Konfiguration der Serverprotokollierung, wie beschrieben, die Standardmethode, um eine ausführliche Protokollierung mit dem `server`-Befehl in WLP zu aktivieren.

---

Hinweis: Seien Sie vorsichtig mit dem `logs`-Verzeichnis. Es enthält Protokolle wie Trace-Dateien.