Action API: Flow Syntax

Grundsätzlich kann Flow überall dort verwendet werden wo auch Action API Blöcke beginnen. Dabei ist es wichtig dass alle Flow Statements in einer Liste geschrieben werden. Hier als Beispiel ein Flow Statement im Trigger Block einer Quest. Wichtig dabei ist, dass alle Flow Statements immer in Hochkomma geschrieben werden. Die Ausnahme bei sind Delays, welche kein Hochkomma benötigen.

Quellcode: quest-flow.quest.yml

```
1. ...
2. trigger:
3. flow:
4. - '@host.interact host:this.example'
5. - '?quest.active quest:this.active-example-quest'
6. - '~2s'
7. - '!conversation.start host:this.example conv:example'
8. ...
```

Inhaltsverzeichnis

In diesem Beispiel kann man auch schon gut die unterschiedlichen Flow Typen erkennen die sich direkt in den Action API Typen widerspiegeln: Trigger, Requirement und Action.

Flow Typen

Je nach dem in welchem Script Block man Flow einsetzt können unterschiedliche Typen verwendet werden. Jeder Flow Typ beginnt mit einem eindeutigen Zeichen und wird von Config Optionen gefolgt.
Folgende Flow Typen stehen dabei zur Auswahl:

- `@`: Trigger, z.B.: `@host.interact host:this.example`
- `!`: Action, z.B.: `!player.tp-coords world:azuran, x:12, y:16, z:-77`
- `?`: Requirement, z.B.: `?player.has-item item:"Robe des Wanderers"`
- `~`: Delay, z.B.: `~2s`, `~1min` oder in Ticks `10`
- `:`: Answer, z.B.: `:"Antwort Möglichkeit 1"` oder eine Antwort mit Input Variable `:"Antwort mit Input"->var1`

Jeder Flow Typ kann dabei, je nach dem in welchem Script Block er verwendet wird, mit anderen Typen kombiniert werden. Dabei kann nicht jeder Flow Typ in jedem Script Block verwendet werden. Antworten z.B. können nur in einem Antwort Block in einer Conversation verwendet werden.
Im folgenden werden die einzelnen Flow Typen noch näher beleuchtet und ihre Kombinations Möglichkeiten aufgezeigt.

Trigger

Trigger können nur in einem Trigger Block, z.B. bei Quests oder Achievements verwendet werden. Einem Trigger können Actions folgen und außerdem kann ein Trigger auch voran gestellte Requirements besitzen die die Ausführung des Triggers weiter einschränken. Jeder Trigger bildet dabei einen eigenen Block mit Actions und die Requirements eines Triggers werden bei jedem aulösen in der entsprechenden Reihenfolge geprüft.

Quellcode: trigger1.yml

```
1. trigger:
2. beispiel1:
3. # der Trigger wird nur ausgelöst wenn der Spieler am Leben ist
4. - '?player.is-alive'
5. # Wenn der Spieler sich in einem Radius von 5 an der Position befindet wird der Trigger ausgelöst
6. # Der Trigger hat einen Cooldown von 1 Minute und wird somit nur einmal in der Minute ausgelöst
7. - '@player.location(cooldown:1min) 128,75,-222,world,5' # genaueres zu der Config Syntax steht weiter unten
```

8. - '!text.warn "Verlasse diesen Ort innerhalb einer Minute und kehre nicht zurück!"'
9. # zähle bei jedem auslösen des triggers eins hoch
10. - '?count(persistant:true, count:2)'
11. # wenn der spieler doch ein zweites mal den Ort betritt wird er getötet
12. - '!player.kill'
13. # verzögere die Ausführung der nächsten Action um 2s
14. - ~2s
15. # sendet dem Spieler eine Nachricht mit dem Namen des NPCs Karl
16. - '!text "Ich habe euch gewarnt!",Karl'
17. # hier beginnt ein neuer Trigger Block und die delays und Requirements von dem vorherigen Block gelten nicht mehr
18. - '@host.interact this.karl'
19. - '!conversation.start host:this.karl, conv:this.example'

Alles anzeigen

Action
Eine Action kann in allen Blöcken verwendet werden und findet somit am meisten Anwendung. Eine Action kann außerdem Requirements voran gestellt haben die nacheinander geprüft werden. In den anderen Beispielen zu Trigger und Requirements gibt es zusätzliche noch einige Beispiele für die Verwendung von Actions.

Quellcode: action1.yml

1. actions:
2. flow:
3. # gibt dem Spieler 5x das Custom Item mit der ID 1337
4. - '!player.give.item rci1337, 5'
5. # prüft ob der spieler die quest example-quest abgeschlossen hat
6. - '?quest.completed ../../example/example-quest'
7. # gibt dem Spieler nur 5000 EXP wenn die "Example" Quest abgeschlossen wurde
8. - '!rcskills.exp.add 5000'
9. # prüft ob der Spieler sich in der aktuellen Welt an den Koordinaten befindet
10. - '?player.location 24,65,-1337'
11. # gibt dem Spieler 1 Gold 22 Silber und 5 Kupfer WENN sich der Spieler an der Position befindet UND die Quest "Example" abgeschlossen hat
12. - '!player.give.money 1g22s5k'

Alles anzeigen

Requirements
Requirements werden vermutlich in den meisten Plugins eingesetzt und haben daher den weitesten Anwendungsbereich. Auch hier kann in allen Plugins die Flow Syntax verwendet werden. In den Action und Trigger Beispielen finden sich bereits genügend Beispiele für die Nutzung. Zu beachten gibt es dabei nur dass man bei `count` Requirements darauf achten muss wie oft und wodurch das Requirements geprüft wird um den richtigen Count Wert zu erhalten.

Delay
Ein Delay kann nur vor Actions verwendet werden und gilt nicht über den Login und Logout eines Spielers hinaus, daher sollten hier am besten nur sehr geringe Werte verwendet werden. Um Actions und Triggern einen längeren `cooldown` zu geben sollte die `cooldown` Variable befüllt werden, z.B.: `@host.interact(cooldown:1d)` oder `!player.kill(cooldown:1y64d10h2m1s10)`. Wenn kein Buchstabe mitgegeben wird ist die Angabe in Ticks.

Beispiel für ein Delay: `~30s`

Answer
Für eine einheitliche Syntax ist es auch möglich Antworten in Conversations 2.0 als Flow anzugeben. Für Input Antworten gibt es dabei eine Spezielle Syntax die eine Variable hinter dem Text erwartet: `:"Antwort Möglichkeit mit Variable"->varname1`.
Eine normale Antwort wird mit Doppelpunkt und in Anführungsstrichen angegeben: `:"Normale Antwort"`. Antworten können Requirements besitzten und von Actions mit Requirements gefolgt werden.

Quellcode: answers1.yml

```
1. answers:
2. flow:
3. - ':"Beispiel Antwort 1"'
4. - '!stage beispiel-1'
5. - ':"Antwort 2 mit Input"->var1'
6. - '!stage "%[var1]"'
7. - '?requirement1'
8. - ':"Antwort mit Requirement"'
```

Flow Config Parser

Jede Flow Action kann beliebige Parameter nach der Typ Definition annehmen. Dabei ist es möglich Positions Parameter und Named Paramter zu mischen, allerdings muss man sich immer bewusst sein dass die Position nur weiter gezählt wird wenn man Positions Paramter verwendet.