Anforderungen - DateV RTS

[**Schnittstellenanforderungen**](#_asyffg7g99h0) **2**

[Task - Model](#_qf3e1x55crs0) 2

[Group - Model](#_l5pk61u5vwty) 3

[Task - API](#_8bs9ogn7dl8m) 4

[Group - API](#_n4ek9qkmohlj) 5

[Management - API](#_lrqq822qy007) 5

[Monitoring - API (Paging)](#_rwdrgl7gx02a) 6

[**Anforderungen an den Real-Time-Scheduler**](#_ecc168b42v70) **6**

[Gruppenhierarchie](#_m0w6m6ti5ag9) 6

[Default Gruppe](#_5fqs11xai7wl) 6

[Prioritäts Berechnung](#_jjuqvfyrosmb) 7

[Dispatcher Anbindung](#_bzgcohskmj8v) 7

[Datenbanken](#_ugdtmownak5f) 7

[**Nicht-funktionale Anforderungen**](#_bef16jk96g4x) **7**

[Cloud-Foundry](#_1zptm19obtxh) 7

[Logging](#_ep5vm3avsgn7) 8

[Sonstiges](#_vhcyva3psy4o) 8

Alles was in dieser Farbe markiert ist weicht von den Anforderungen ab

# Schnittstellenanforderungen

## Task - Model

|  |  |
| --- | --- |
| id\* | * Das Attribut id ist ein String der jeden Task eindeutig identifiziert |
| groupId | * Die groupID gibt den Namen der Gruppe wieder, die der Task zugehörig ist * Default: Default Gruppe |
| priority | * Das Attribut priority gibt die relative Priorität des Task an als Integer Wert an * Der Wert von priority liegt zwischen Null und 9999, wobei Null die höchste Priorität darstellt * Default wird gesetzt durch die Default Gruppe |
| deadline | * Das Attribut deadline gibt den spätesten Zeitpunkt an bis der Task abgeschlossen sein muss * Je näher ein Task an der deadline liegt je höher wird seine interne Priorität |
| active\_times | * Das Attribut active\_times gibt die Zeitabschnitte an, an denen der Task bearbeitet werden kann * Die Zeitabschnitte werden als Aneinanderkettung der jeweiligen Zeitabschnitte als String Format übergeben |
| working\_days | * Das Attribut working\_days gibt die Wochentage an, an denen der Task bearbeitet werden kann * Die Wochentage werden als Boolean-Array der Länge sieben übergeben, wobei jeder Boolean-Wert für einen Wochentag steht, beginnend mit Montag |
| status | * Der Wert status gibt an in welchem Status sich der jeweilige Task befindet (“waiting”, “scheduled” “dispatched” and “finished”) * Beim Erstellen eines neuen Tasks wird der Status auf “waiting” gesetzt |
| type\_flag | * Das Attribut type\_flag gibt an ob es sich um einen “batch” oder “realtime” Auftrag handelt * Ein realtime Auftrag wird von Personen erstellt und haben deshalb eine höhere Priorität * Batch Aufträge stammen von Programmen und haben eine niedrigere Priorität * Default wird gesetzt durch die Default Gruppe |
| mode | * Das Attribut mode beschreibt, ob es sich um einen sequentiellen oder parallelen Auftrag handelt * Bei einem sequentiellen Auftrag muss eine bestimmte Reihenfolge eingehalten werden * Bei einem parallelen Auftrag ist die Reihenfolge egal * Default wird gesetzt durch die Default Gruppe |
| retries | * Das Attribut retries gibt an, wie viele Fehlschläge bei der Bearbeitung von einem Task versucht werden, bevor seine interne Priorität verringert wird * Zählt hoch |
| force | * Wenn der Parameter force gesetzt wird, erfolgt eine direkte Bearbeitung des Auftrags * Default: false |
| index\_number | * Das Attribut index\_number gibt den Index in der Reihenfolge der Bearbeitung an, falls es sich um einen sequentiellen Auftrag handelt |
| meta\_data | * Das Attribut meta\_data ist ein String mit Zusatzinformationen, die durchgereicht werden |
| history | * Beinhaltet Timestamps des Tasks, zu denen er seinen Status gewechselt hat im JSON Format * Format der Timestamps: “Milliseconds since epoch” |

*\*Pflichtfeld*

## 

## Group - Model

|  |  |
| --- | --- |
| id\* | * Das Attribut id ist ein String der jede Gruppe eindeutig identifiziert. * not nullable |
| parent\_id | * Das Attribut parent\_id gibt die Id der übergeordneten Gruppe an |
| priority | * Das Attribut priority gibt den Defaultwert der Gruppe für alle Tasks an * Default: 1 |
| deadline | * Das Attribut deadline gibt den Defaultwert der Gruppe für alle Tasks an |
| active\_times | * Das Attribut active\_times gibt den Defaultwert der Gruppe für alle Tasks an |
| working\_days | * Das Attribut working\_days gibt die Defaultwert der Gruppe für alle Tasks an * Die Wochentage werden als Boolean-Array der Länge sieben übergeben, wobei jeder Boolean-Wert für einen Wochentag steht, beginnend mit Montag |
| type\_flag | * Das Attribut type\_flag gibt den Defaultwert der Gruppe für alle Tasks an * Default: Parallel |
| mode | * Das Attribut mode gibt den Defaultwert der Gruppe für alle Tasks an * Default: Batch |
| last\_index\_number(\*) | * Das Attribut last\_index\_number gibt den Index des zuletzt bearbeiteten Auftrags in einer sequentiellen Abarbeitung an. * Verpflichtend, wenn type\_flag auf Sequentiell gesetzt wird |
| parallelism\_degree(\*) | * Das Attribut parallelism\_degree gibt den Defaultwert der Gruppe für alle Tasks an. * Verpflichtend, wenn type\_flag auf Parallel gesetzt wird |

*\*Pflichtfelder*

## 

## Task - API

|  |  |
| --- | --- |
| Create | * Task wird erstellt und gescheduled * Startet im Status “waiting” |
| Read | * Task wird als JSON zurückgegeben |
| Update | * Task Attribute können geändert werden * Task wird daraufhin neu gescheduled |
| Delete | * Task wird gelöscht |
| Batch create | * Wie Create nur kann eine Liste an Tasks mitgegeben werden |

## 

## Group - API

|  |  |
| --- | --- |
| Create | * Gruppe wird erstellt |
| Read | * Gruppe wird als JSON zurückgegeben |
| Update | * Gruppen Attribute können geändert werden * Alle Tasks werden daraufhin neu gescheduled |
| Delete | * Gruppe wird gelöscht * Schlägt fehl, wenn Gruppe Kinder hat (HTTP 400) |

## Management - API

|  |  |
| --- | --- |
| Pausieren/Fortsetzen von Gruppe | * Alle Untergruppen und Tasks werden pausiert/fortgesetzt |
| Pausieren/Fortsetzen von Task | * Task wird pausiert/fortgesetzt * Task mit “force” wird trotzdem dispatched |
| Selektierte Tasks sofort starten | * Tasks werden sofort ausgeführt (Wurde mit Einführung des force Attributs in seiner damaligen Implementierung obsolet) |
| Deadline für Fortsetzung | * Ein mit gegebener Timestamp gibt an, bis wann ein Task/Gruppe pausiert werden soll * Cleaning Service scannt in einem konfigurierbaren (application.properties) Intervall Tasks/Gruppen deren Timestamp zu fortsetzen erreicht wurde * Über die properties kann eingestellt werden in welchem Intervall gecheckt werden soll ob pausierte Tasks fortgesetzt werden können |
| Pausieren/Fortsetzen des RTS | * Anfragen werden noch entgegengenommen * Es wird gescheduled * Dispatchment wird pausiert * Task mit “force” wird trotzdem dispatched |

* In den Properties des Cleaner Service kann ein Intervall konfiguriert werden, welches angibt, wie oft gecheckt werden soll ob pausierte Tasks/Gruppen wieder fortgesetzt werden sollen (Default: Einmal in der Minute)
* In den Eigenschaften des Cleaner Service kann ein Intervall (Default: Einmal am Tag) und eine Anzahl an Tagen konfiguriert werden. Diese geben an in welchem Intervall Tasks im Status “finished” und älter als die konfigurierten Tage (Default: 10 Tage) endgültig gelöscht werden

## Monitoring - API (Paging)

|  |  |
| --- | --- |
| Timestamps | * Die Timestamps und IDs aller Tasks für die Ereignisse: Annahme, Weitergabe und Feedback werden ausgegeben |
| Count RTS | * Alle Tasks werden gezählt nach dem Status in dem sie sich befinden (Noch nicht eingeplant, eingeplant, in Bearbeitung, abgeschlossen und fehlgeschlagen) * ID der Tasks |
| Count Gruppe | * Tasks einer Gruppe werden gezählt nach dem Status in dem sie sich befinden (Noch nicht eingeplant, eingeplant, in Bearbeitung, abgeschlossen und fehlgeschlagen) * ID der Tasks |

* In der API Specification sind diese Schnittstellen definiert aber noch nicht implementiert:
  + groupCountGet
  + taskDispatchedErrorsGet
  + taskDispatchedGet
  + taskScheduledGet
  + taskWaitingGet

# Anforderungen an den Real-Time-Scheduler

## Gruppenhierarchie

* Task hat immer genau eine Parent Gruppe (Default wenn nicht angegeben)
* Gruppe kann auch genau eine direkte Parent Gruppe haben
* Ist bei einem Task das Feld “active\_times” nicht gesetzt wir von unten die Hierarchie durchlaufen, bis eine Gruppe erreicht wird, bei der die “active\_times” gesetzt sind. Diese werden dann genutzt
* Eine Gruppe soll entweder **nur Tasks** oder **nur Gruppen** als Kinder haben

### Default Gruppe

|  |  |
| --- | --- |
| **Attribut** | **Wert** |
| id | 'DEFAULT\_GROUP' |
| priority | 1 |
| type\_flag | 'Batch' |
| mode | 'Parallel' |
| parallelism\_degree | 100 |

## Prioritäts Berechnung

* Priorität ist von 1 aus absteigend (1: Höchste Priorität, 9999: niedrigste Priorität)
  + intern wird andersrum priorisiert
* Je näher das Datum im Feld “deadline” rückt, desto höher soll der Task priorisiert werden
* Wenn bei einem Task das Feld “force” auf true gesetzt wird soll der Task sofort an die Dispatcher Instanz weitergeleitet werden
  + Die Reihenfolge bei “force” Tasks muss nicht gewährleistet sein
  + First-In-First-Out (FIFO)
* Je öfter ein Task fehlschlägt, desto niedriger wird seine Priorität (Wird nicht in die Prioritäts Berechnung einbezogen)

## Dispatcher Anbindung

* Ausgang des RTS als Interface (Methode write)
  + Wir implementieren RabbitMQ etc
  + DATEV als IBM-MQ
  + Soll über Spring konfigurierbar sein (Profile müsste hinzugefügt werden)
* Eingang des RTS als Interface (Methode read)
* Feedback kommt, wenn Auftrag komplett fertig bearbeitet wurde
  + Fehlschlagen wird von der DATEV gehandelt
* Feedback Annahme ist notwendig um den Parallelitätsgrad aktuell zu halten

## Datenbanken

* Verwaltet persistent die zu "schedulenden" Tasks und Gruppen
* Muss robust sein
* Cleanup Intervall für Tasks die älter als X Tage sind und Feedback bekommen haben soll konfigurierbar sein

# Nicht-funktionale Anforderungen

## Cloud-Foundry

* Markdown decision für Services/Technologien die nicht verwendet wurden
* Step-by-step Anleitung wie die Dienste, die der Real-Time-Scheduler nutzt, Angebunden und konfiguriert werden müssen

## Logging

* Format **-> ist etwas anders, taskid und groupid = X gibt es bei uns nicht, wird aber beim logger am anfang mitgegeben also z.B. “taskid: 123, dispatched”**
  + date=X level=X message=X
  + date=X level=X taskid=X message=X (mit Bezug auf Task)
  + date=X level=X groupid=X message=X (mit Bezug auf Group)
  + Format weicht ab:
    - 2020-07-14 14:14:48,617 INFO - Task 4da9542f-d59f-4628-9932-9e2ed0ac3ab5 received
    - 2020-07-14 14:14:48,617 INFO - Group TestGroup1 received
* Loglevel
  + ERROR: Jede Exception, die geworfen wird und zum Abbruch führen kann
  + WARN: Jede Exception, die gefangen wird aber weitergearbeitet werden kann
  + INFO: Wichtige Prozessschritte, bspw. Programmstart, Ausführungszeiten, Auftrags-/Auftragsstatus
  + Es gibt Fälle in denen Exceptions auf INFO Level geloggt werden
  + Beispiel: Transaktion schlägt fehl: Logger Level INFO wird genutzt
  + DEBUG: Grober Programmablauf, Methodenaufruf und Parameter (darf nicht nach Splunk geschrieben werden)
* Spring Transaktions Logging soll konfigurierbar sein

## Sonstiges

* Lower-snake-case für Model Attribute verwenden