# *JiffyBox-API*

Dokumentation zu Version 1.0

# Inhaltsverzeichnis

 

# Einführung

CloudServer On Demand von JiffyBox sind virtuelle Server, deren Ressourcen nicht fix sind, sondern aus einem großen Pool, der „Cloud“ kommen. Ohne starres Preiskorsett und ohne Mindestlaufzeiten erfolgt die Abrechnung ausschließlich nutzungsbasiert. Weitergehende Informationen zu den Features und Preisen von JiffyBoxen finden Sie auf unserer Webseite:

- Für Kunden aus Deutschland: http://www.df.eu/de/cloud-hosting/cloud-server/
- Für Kunden aus Österreich: http://www.df.eu/at/cloud-hosting/cloud-server/

Während das Control-Panel bereits eine einfache und schnelle Bedienung von JiffyBox ermöglicht, kann sich das volle Potential vieler Funktionen, wie beispielsweise das Einfrieren und Auftauen, insbesondere durch die maschinelle, geskriptete Nutzung entfalten. Neben dem Control-Panel steht für JiffyBox daher eine Schnittstelle zur Programmierung (API) zur Verfügung, um Ihnen die größtmögliche Freiheit bei der Arbeit mit virtuellen Servern von JiffyBox zu ermöglichen.

Die Schnittstelle stellt einen sehr großen Teil der auch über das Control-Panel verfügbaren Operationen zur Verfügung. Lediglich im Bereich der Antwort-Codes asynchroner Nachrichten kann es vorkommen, dass derzeit noch kein identischer Funktionsumfang erreicht ist.

Diese Dokumentation richtet sich primär an ambitionierte Software-Entwickler, die basierend auf unserer Schnittstelle eigene Applikationen entwickeln möchten. Es wird vorausgesetzt, dass der Leser ein grundlegendes Verständnis der folgenden Technologien hat:

- HTTP-Protokoll in Version 1.1
- RESTful Web Services
- JSON

Alle Beispiele sind komplett in JSON gehalten. Eine zukünftige Unterstützung anderer Protokolle (XML, SOAP etc.) ist grundsätzlich denkbar. Sofern Sie Bedarf für andere Protokolle haben, würden wir uns über Ihr Feedback in unserem Forum freuen: https://forum.df.eu

# Begriffe

In diesem Kapitel werden einige Begriffe definiert, die für die Nutzung dieser Dokumentation wichtig sind.

## Host

Der Host ist ein physikalischer Rechner, der seine Hardware in Form einzelner Instanzen, den JiffyBoxen, zur Verfügung stellt. Je nach Ausstattung kann ein Host eine bestimmte Anzahl von Tarifen zur Verfügung stellen.

## JiffyBox

JiffyBox ist eine Instanz auf einem Host. Das bedeutet, dass eine JiffyBox sich wie ein eigener Computer nach außen darstellt und zwar genau so, wie sie vorher im Control-Panel konfiguriert wurde (Festplatten, Kernel etc.).

## Festplatte

Die Daten der JiffyBox werden auf Festplatten abgelegt. Jede JiffyBox kann über eine oder mehrere Festplatten verfügen. Standardmäßig legt das System für jede JiffyBox beim Anlegen eine Festplatte für das Dateisystem und eine für Swap an. Festplatten müssen aus der JiffyBox mittels der Device-Namen /dev/xvda, /dev/xvdb etc. angesprochen werden.

## Profil

Profile stellen die komplette Konfiguration einer JiffyBox und damit eines virtuellen Computers dar. Sie beschreiben den zu bootenden Linux-Kernel, die Festplatten, die Runlevel etc. Ohne ein aktives Profil kann eine JiffyBox nicht booten.

## Backups

Backups sind Abbilder (Images) der Festplatten einer JiffyBox zu einem bestimmten Zeitpunkt. Beim Erstellen eines Backups werden von allen Festplatten der JiffyBox hintereinander Schnappschuss-Kopien erstellt und diese auf ein externes Storage gesichert.

# Grundlagen

Die JiffyBox-API nutzt ein RESTful Web Services Interface. Das bedeutet, dass jedes Kommando eindeutig identifizierbar und zustandslos ist. Sie können die Verbindung also jederzeit trennen und später ein neues Kommando schicken. Es gibt keine zeitkritischen Vorgänge und es kann bei jeder Anfrage auch ein anderer API-Server antworten.

Die Nutzung eines RESTful Web Service Interfaces bedeutet zudem, dass neben `POST`- und `GET`-Requests auch `DELETE` und `PUT` benötigt werden.

Um die Arbeit mit der API angenehm zu gestalten, können Parameter bei `PUT`- oder `POST`-Operationen entweder in der bekannten `key1=value1&key2=value2`-Schreibweise übergeben werden, oder wahlweise auch einfach als JSON-Paket der Form `{"key1":value1,"key2":value2}`.

## Die URL

Jeder Aufruf der API besteht aus einer URL des folgenden Formats:

```
  /<modul>[/<bereich>][/<bereich N>]
```

Nicht jedes Modul hat einen eigenen Bereich und manche haben mehr als einen Bereich. Wie ein Modul aufgerufen werden kann, können Sie der jeweiligen Dokumentation des Moduls entnehmen.

## Authentifizierung

Um mit der API zu kommunizieren, müssen Sie sich zunächst authentifizieren. Hierfür benötigen Sie den sogenannten „API-Token“, den Sie im Control-Panel unter „Account“, „API-Zugriff“ erstellen können. Dieser Token muss in der URL bei jedem API-Aufruf mitgegeben werden.

Jeder Kunden-Account kann zu jeder Zeit immer nur genau einen API-Token besitzen. Der Token ermöglicht einen einfachen und sicheren Zugriff auf die API. Die API ist zudem ausschließlich verschlüsselt über SSL ansprechbar, so dass eine hohe Sicherheit gewährleistet ist.

Der API-Token ist 32 Zeichen lang und hat eine unbegrenzte Lebensdauer. Sie können die Gültigkeit allerdings selbst entziehen, indem Sie den Token löschen oder einen neuen erzeugen.

## Limits

Um die API-Server nicht zu überlasten, ist die Anzahl der Aufrufe limitiert. Jedes Modul kann zwar eine eigene Aufrufs-Limitierung definieren, jedoch gelten 2 Limitierungen global:

| | |
|---|---|
| **Login-Versuche** | 1 Fehler pro Sekunde, 5 Fehler pro Minute |
| **API-Calls** | 30 Aufrufe pro Minute |

Ein Verstoß gegen ein Limit erhöht das Limit  nicht. Wenn Sie also z.B. 10 falsche Logins in der ersten Sekunde schicken, wird nur einen Fehlversuch berücksichtigt, da die 9 weiteren vom Ratelimit abgelehnt werden.

# API-Versionierung

Das Internet entwickelt sich ständig weiter und so ändern sich auch die Schnittstellen für Anwendungen. Diese API hat aktuell die Version 1.0. Immer dann, wenn wir einen Aufruf komplett ändern müssen, so dass ein alter Aufruf Fehler brächte, werden wir die API-Version erhöhen. Innerhalb einer API-Version sind alle Neuerungen und Änderungen abwärtskompatibel. Wenn dies einmal nicht möglich ist, muss eine neue Version der API genutzt werden. Die API-Version, die von einem Aufruf genutzt wird, ist immer Teil der URL.

# Rückgabewerte

Jeder Aufruf liefert einen JSON-kodierten Hash zurück, der Meldungen sowie den Rückgabewert enthält. Ein Beispiel dafür ist:

```
{
  "messages": [],
  "result": {
    "doc": "Das Modul doc",
    "distributions": "Modul zum Auflisten von installierbaren Linux-
                      Distributionen",
    "jiffyBoxes": "Modul zum Auflisten und Manipulieren von
                   JiffyBoxen",
    "backups": "…",
    "plans": "…"
  }
}
```

Dabei sind `messages` und `result` immer vorhanden. `messages` sehen z.B. wie folgt aus:

```
{
  "messages": [
    {
      "type": "error",
      "message": "Das Modul test existiert nicht"
    }
  ],
  "result": false
}
```

# Fehler

Fehler bzw. „messages“ haben konkrete Typen: `error`, `warning`, `notice` und `success`.

Ein Rückgabewert von „false“ signalisiert einen Fehler und sollte immer eine entsprechende Nachricht mitbringen. Es können allerdings auch Nachrichten kommen, wenn kein Fehler auftritt. In diesem Fall handelt es sich um rein informelle Hinweise (z.B. Auftauen einer nicht eingefrorenen JiffyBox etc.).

# Verben

Als Verben werden die HTTP-Methoden bezeichnet, die zum Zugriff auf eine Funktionalität erforderlich sind. Dabei gilt generell:

| | |
|---|---|
| **GET** | Zum Abfragen von Informationen |
| **POST** | Zum Neuanlegen eines Objektes oder sonstigen Operationen |
| **PUT** | Zum Manipulieren eines Objektes |
| **DELETE** | Zum Löschen eines Objektes |

Die Methode eines HTTP-Requests lässt sich bequem mit dem Tool Curl (http://curl.haxx.se/) als Option -X übergeben. Beispiele:

```
curl https://api.jiffybox.de/Token/v1.0/jiffyBoxes/12345
```

Zeigt Informationen über JiffyBox 12345 per GET-Request.

```
curl -X DELETE https://api.jiffybox.de/Token/v1.0/jiffyBoxes/12345
```

Löscht die JiffyBox 12345.

Wird bei Curl -d als Option übergeben, impliziert dies einen POST-Request:

```
curl -d name=Test -d planid=10
https://api.jiffybox.de/Token/v1.0/jiffyBoxes/12345
```

Erstellt eine Kopie der JiffyBox Nr. 12345 mit Namen Test im Tarif CloudLevel 1.

Wollen Sie stattdessen einen PUT-Request, müssen Sie dies explizit angeben:

```
curl -X PUT -d status=START
https://api.jiffybox.de/Token/v1.0/jiffyBoxes/12345
```

Modifiziert die JiffyBox Nr. 12345 so, dass ihr Status `RUNNING` ist. Bootet die JiffyBox also.

Die selbe Ressource kann, wie in den Beispielen gut sichtbar, mit verschiedenen Verben komplett anders angesprochen und manipuliert werden. **Achten Sie deshalb beim Lesen der Dokumentation immer genau darauf, welches Verb Sie benötigen**.

# Parameter

Die einzelnen Kommandos haben optionale Parameter und Pflichtparameter. Pflichtparameter müssen übergeben werden, da sonst ein entsprechender Fehler zurückgegeben wird. Optionale Parameter können übergeben werden, müssen aber nicht. In den Tabellen sind die Parameter wie folgt gegliedert:

| | |
|---|---|
| **Parameter 1** | Pflichtparameter |
| *Parameter 2* | Optionaler Parameter |

# Beispiele

Die Beispiele zeigen optisch aufgewertet konkrete Parameter und Rückgabewerte von Beispielaufrufen. Um zwischen Aufruf und Rückgabe zu unterscheiden, ist der Aufruf fett und der Rückgabewert normal formatiert.


# curl https://api.jiffybox.de/<Token>/v1.0/eineURL
Aufruf
{
"messages": [],
"result" : {}
}
Rückgabewert

# API-Kommandos

## JiffyBox-Kommandos

### Alle JiffyBoxen auflisten

Verb: **GET** URL: **/jiffyBoxes**

### Beschreibung

Mit diesem Aufruf erhalten Sie eine Übersicht aller eingerichteten JiffyBoxen. Da dieser Aufruf sehr zeitintensiv ist, raten wir von der Verwendung zur kontinuierlichen Abfrage des Status explizit ab. Hierfür sollten Sie den spezifischeren Aufruf auf die einzelne JiffyBox nutzen.

### Beispiel

```
# curl https://api.jiffybox.de/<token>/v1.0/jiffyBoxes
```


```
{
  "messages": [],
  "result": {
    "12345": {
      "id": 12345,
      "name": "Test",
      "ips": {
        "public": ["188.93.14.176"],
        "private": ["10.93.14.175"]
      },
      "status": "READY",
      "created": 1234567890,
      "recoverymodeActive": false,
      "manualBackupRunning": false,
      "isBeingCopied": false,
      "running": false,
      "host": "vmhost-testsys-2-2-9-1",
      "plan": {
    "id":22,
    "name":"CloudLevel 3",
    "diskSizeInMB":307200,
    "ramInMB":8192,
    "pricePerHour":0.07,
    "pricePerHourFrozen":0.02,
    "cpus":6
      },
      "metadata": {
        "createdby": "JiffyBoxTeam"
      },
      "activeProfile": {
        "name": "Standard",
        "created": 1234567890,
        "runlevel": "default",
        "kernel": "xen-current",
        "rootdisk": "\/dev\/xvda",
        "rootdiskMode": "ro",
        "status": "READY",
        "disks": {
          "xvda": {
```

```
            "name": "CentOS 5.4",
            "filesystem": "ext3",
            "sizeInMB": 81408,
            "created": 1234567890,
            "status": "READY",
            "distribution": "centos_5_4_32bit"
          },
          "xvdb": {
            "name":" Swap",
            "filesystem": "swap",
            "sizeInMB": 512,
            "created": 1234567890,
            "status": "READY"
          }
        }
      }
    }
  }
}
```

# Details zu einer JiffyBox abfragen

Verb: **GET** URL: **/jiffyBoxes/<Box-ID>**

## Beschreibung

Liefert Details zu einer bestimmen JiffyBox. Der Rückgabewert entspricht dem einer einzelnen Box in `/jiffyBoxes`. Wenn Sie mehr als nur eine JiffyBox haben, ist die Auslesegeschwindigkeit allerdings um ein Vielfaches höher.

## Beispiel

```
# curl https://api.jiffybox.de/<token>/v1.0/jiffyBoxes/12345
```


```
{
  "messages": [],
  "result": {
    "id": 12345,
    "name": "Test",
    "ips": {
      "public": ["188.93.14.176"],
      "private": ["10.93.14.175"]
    },
    "status": "READY",
    "created": 1234567890,
    "recoverymodeActive": false,
    "manualBackupRunning": false,
    "isBeingCopied": false,
    "running": false,
    "host": "vmhost-testsys-2-2-9-1",
    "plan": {
      "id": 22,
      "name": "CloudLevel 3",
      "diskSizeInMB": 307200,
      "ramInMB": 8192,
      "pricePerHour": 0.07,
      "pricePerHourFrozen": 0.02
      "cpus":6
    },
    "metadata": {
     "createdby": "JiffyBoxTeam"
   },
   "activeProfile": {
      "name": "Standard",
      "created": 1234567890,
      "runlevel": "default",
      "kernel": "xen-current",
      "rootdisk": "\/dev\/xvda",
      "rootdiskMode": "ro",
      "status": "READY",
      "disks": {
        "xvda": {
          "name": "CentOS 5.4",
          "filesystem": "ext3",
          "sizeInMB": 81408,
          "created": 1234567890,
          "status": "READY",
          "distribution": "centos_5_4_32bit"
        },
```


 

```
        "xvdb": {
          "name":" Swap",
          "filesystem": "swap",
          "sizeInMB": 512,
          "created": 1234567890,
          "status": "READY"
        }
      }
    }
  }
}
```

# Eine JiffyBox löschen

Verb: **DELETE** URL: **/jiffyBoxes/<Box-ID>**

## Beschreibung

Mit diesem Kommando können Sie JiffyBoxen löschen. Das Kommando ist asynchron. Bei Erfolg wird `true` geliefert, bei einem Fehler `false`. Ein Erfolg bedeutet lediglich, dass das Löschen erfolgreich beantragt wurde. Das eigentliche Löschen findet asynchron statt. Der Status der Löschung kann durch Auslesen der JiffyBox-Daten erfragt werden. Sobald die Meldung kommt, dass die JiffyBox nicht mehr existiert, ist der Löschvorgang abgeschlossen. Während des Löschvorgangs ist der Status der JiffyBox `DELETING`. Sollte dieser wieder auf `READY` springen, war der Löschvorgang nicht erfolgreich.

## Beispiel

```
# curl -X DELETE https://api.jiffybox.de/<token>/v1.0/jiffyBoxes/12345
```

```
{
  "messages": [],
  "result": true
}
```

# Eine JiffyBox neu erstellen

Verb: **POST** URL: /**jiffyBoxes**

## Beschreibung

Es gibt 3 Wege, eine JiffyBox zu erstellen:

1. Die JiffyBox komplett neu erstellen
2. Sie als Kopie einer vorhandenen JiffyBox erstellen
3. Sie aus einem Backup einer JiffyBox erstellen

Dieses Kommando behandelt Weg 1 und 3 und benötigt einige Parameter, um zu funktionieren. Das Erstellen einer JiffyBox ist ein asynchroner Vorgang. Ein Erfolg als Rückgabe bedeutet nur, dass das Erstellen gestartet wurde. Über das Ergebnis kann nur ein periodisches Abfragen der JiffyBox-Daten via `GET`-Request auf `/jiffyBoxes/<id>` Auskunft geben. `id` ist in den Rückgabewerten der Erstellung enthalten. Während des Erstellens ist der JiffyBox-Status auf `CREATING`. Wenn die JiffyBox fertig eingerichtet ist, wechselt der Status bei Erfolg auf `READY`. Bei Misserfolg wird die JiffyBox automatisch wieder gelöscht.

## Parameter

| | |
|---|---|
| **name** | Der Name der neu zu erstellenden JiffyBox. Erlaubt sind bis zu 30 Zeichen. Erlaubte Zeichen sind: `a-zA-Z0-9üöäÜÖÄß_()=!\*@.-` und Leerzeichen. Der Name der JiffyBox darf nicht doppelt vergeben werden, sonst wird eine entsprechende Fehlermeldung ausgegeben. |
| **planid** | Der Plan bzw. Tarif einer JiffyBox ist entweder die eindeutige Nummer oder aber der Name des Tarifs (`CloudLevel 1`, `CloudLevel 2` etc.). Die IDs und weitere Details der Tarife können unter `/plans` abgefragt werden. |
| *backupid* | Wenn übergeben, gibt die Backup-ID die ID des Backups an, aus dem eine neue JiffyBox erstellt werden soll. Eine Liste aller Backup-IDs können Sie über die URL `/backups` auslesen. Dieser Parameter und `distribution` schließen sich gegenseitig aus. Es muss jedoch immer genau einer der beiden übergeben werden. |
| *distribution* | Wenn übergeben, gibt dieser Parameter an, mit welcher Distribution die Festplatte initialisiert werden soll. Eine Liste aller verfügbaren Distributionen kann über die URL `/distributions` ausgelesen werden. Dieser Parameter und `backupid` schließen sich gegenseitig aus. Es muss jedoch immer genau einer der beiden übergeben werden. |
| *password* | Wenn das Passwort nicht zufällig vergeben werden soll, kann mit diesem Parameter explizit eines gesetzt werden. Wird der Parameter nicht angegeben, wird automatisch ein Passwort generiert und an die hinterlegte E-Mail-Adresse gesendet. |

| | |
|---|---|
| *use_sshkey* | Wenn dieser Parameter übergeben wird und im Control-Panel ein SSH-Key hinterlegt wurde, wird diesem SSH-Key nach dem automatischen Einrichten Root-Zugang ohne Passwort gewährt. |
| *metadata* | Zu jeder JiffyBox können optional Daten hinterlegt werden, welche dann bei jedem Auslesen wiedergegeben werden. Metadaten werden als Hash übergeben, um ein Höchstmaß an Flexibilität zu erreichen. Der Wert selbst kann ein beliebig komplexer Typ sein (Hash/Objekt, Array, Boolean etc.) und wird genau so wiedergegeben, wie er übergeben wurde. Insgesamt dürfen die gespeicherten Metadaten bis zu 4 kb belegen. Bitte beachten Sie, dass der Platzbedarf sich auf den aus den Metadaten generierten JSON-String bezieht. |

## Beispiele

```
# curl -d name=Test -d planid=10 -d distribution=centos_5_4_64bit \
       https://api.jiffybox.de/<Token>/v1.0/jiffyBoxes
```


```
{
  "messages": [],
  "result": {
    "id": 12345,
    "name": "Test",
    "ips": {
      "public": ["188.93.14.211"],
      "private": ["10.93.14.211"]
    },
    "status": "CREATING",
    "created": 1234567890,
    "recoverymodeActive": false,
    "manualBackupRunning": false,
    "isBeingCopied": false,
    "running": false,
    "host": "vmhost-testsys-2-2-9-2",
    "plan": {
      "id":20,
      "name": "CloudLevel 1",
      "diskSizeInMB": 76800,
      "ramInMB": 2048,
      "pricePerHour": 0.02,
      "pricePerHourFrozen": 0.005
      "cpus": 3
    }
  }
}
```


```
# curl -d name=Test -d planid=10 \
       -d backupid=1234567890abcdef1234567890abcdef \
       https://api.jiffybox.de/<Token>/v1.0/jiffyBoxes
```


```
{
  "messages": [],
  "result": {
    "id": 12346,
    "name": "Test",
    "ips": {
      "public": ["188.93.14.212"],
      "private": ["10.93.14.212"]
    },
```

```
    "status": "CREATING",
    "created": 1234567890,
    "recoverymodeActive": false,
    "manualBackupRunning": false,
    "isBeingCopied": false,
    "running": false,
    "host": "vmhost-testsys-2-2-9-2",
    "plan": {
      "id":20,
      "name": "CloudLevel 1",
      "diskSizeInMB": 76800,
      "ramInMB": 2048,
      "pricePerHour": 0.02,
      "pricePerHourFrozen": 0.005
      "cpus": 3
    }
  }
}
```

Bei der Nutzung von Metadaten empfiehlt es sich, JSON-Pakete zu nutzen:

```
# curl -d '{"name": "Test", "planid": 10,
            "distribution": "centos_5_4_64bit",
            "metadata": {"createdBy": "The JiffyBoxTeam",
                         "usedBy": ["Me", "You", "Everyone"],
                         "freeForAll": false}}'
       https://api.jiffybox.de/<Token>/v1.0/jiffyBoxes
```

```
{
  "messages": [],
  "result": {
    "id": 12345,
    "name": "Test",
    "ips": {
      "public": ["188.93.14.211"],
      "private": ["10.93.14.211"]
    },
    "status": "CREATING",
    "created": 1234567890,
    "recoverymodeActive": false,
    "manualBackupRunning": false,
    "isBeingCopied": false,
    "running": false,
    "host": "vmhost-testsys-2-2-9-2",
    "plan": {
      "id":20,
      "name": "CloudLevel 1",
      "diskSizeInMB": 76800,
      "ramInMB": 2048,
      "pricePerHour": 0.02,
      "pricePerHourFrozen": 0.005
      "cpus": 3
    }
  },
  "metadata": {
    "createdBy": "The JiffyBoxTeam",
    "usedBy": ["Me", "You", "Everyone"],
    "freeForAll": false
  }
}
```

 

# Eine JiffyBox duplizieren

Verb: **POST** URL: **/jiffyBoxes/<Box-ID>**

## Beschreibung

Mit diesem Kommando können Sie eine neue JiffyBox als Kopie einer bestehenden erstellen. Dabei gelten ebenfalls die Hinweise aus dem Kapitel zum Erstellen einer neuen JiffyBox.

## Parameter

| | |
|---|---|
| **name** | Der Name der neu zu erstellenden JiffyBox. Erlaubt sind bis zu 30 Zeichen. Erlaubte Zeichen sind: `a-zA-Z0-9üöäÜÖÄß_()=!\*@.-` und Leerzeichen. Der Name der JiffyBox darf nicht doppelt vergeben werden, sonst wird eine entsprechende Fehlermeldung ausgegeben. |
| **planid** | Der Plan bzw. Tarif einer JiffyBox ist entweder die eindeutige Nummer oder aber der Name des Tarifs (`CloudLevel 1`, `CloudLevel 2` etc.). Die IDs und weitere Details der Tarife können unter `/plans` abgefragt werden. |
| *metadata* | Zu jeder JiffyBox können optional Daten hinterlegt werden, welche dann bei jedem Auslesen wiedergegeben werden. Metadaten werden als Hash übergeben, um ein Höchstmaß an Flexibilität zu erreichen. Der Wert selbst kann ein beliebig komplexer Typ sein (Hash/Objekt, Array, Boolean etc.) und wird genau so wiedergegeben, wie er übergeben wurde. Insgesamt dürfen die gespeicherten Metadaten bis zu 4 kb belegen. Bitte beachten Sie, dass der Platzbedarf sich auf den aus den Metadaten generierten JSON-String bezieht. |

## Beispiel

```
curl -d name=Test -d planid=10
https://api.jiffybox.de/<Token>/v1.0/jiffyBoxes/12345
```


```
{
  "messages":[],
  "result":  {
    "id":40978,
    "name":"Test",
    "ips":  {
      "public":["93.180.154.7"],
      "private":["10.1.16.179"]
    },
    "status": "CREATING",
    "created":1359635993,
    "recoverymodeActive":false,
    "manualBackupRunning":false,
    "isBeingCopied":false,
    "running":false,
    "host":"vmhost-2-2-8-11",
    "plan":  {
      "id":20,
      "name":"CloudLevel 1",
      "diskSizeInMB":76800,
      "ramInMB":2048,
```


 

```
      "pricePerHour":0.02,
      "pricePerHourFrozen":0.005,
      "cpus":3
    },
    "metadata":{
    }
  }
}
```


 

# Eine JiffyBox starten, stoppen, einfrieren und auftauen

Verb: **PUT** URL: **/jiffyBoxes/<Box-ID>**

## Beschreibung

Um den Zustand einer JiffyBox zu ändern, ist ein PUT-Request nötig, der den neuen Status setzt. Das zu modifizierende Feld ist immer `status`. Dieses Kommando wird komplett asynchron durchgeführt.

## Parameter

| | |
|---|---|
| **Status** | Der einzutragende, neue Status. Mögliche Werte sind: `START`, `SHUTDOWN`, `PULLPLUG`, `FREEZE` und `THAW`. Direkt nach dem Aufrufen ändert sich das Feld `status` auf `UPDATING`, `FREEZING` oder `THAWING`, um anzuzeigen, dass sich `status` ändert. So lange `status` auf diesen Werten steht, akzeptiert die JiffyBox keine weiteren Kommandos. Bei Erfolg ändert sich `status` anschließend auf `READY` (Ausnahme: bei `FREEZE` auf `FROZEN`), um anzuzeigen, dass die JiffyBox jetzt bereit ist, weitere Kommandos entgegenzunehmen. Ob die Operationen `START`, `SHUTDOWN` oder `PULLPLUG` erfolgreich waren, erkennt man daran, dass sich das Feld `running` auf `TRUE`, bzw. `FALSE` ändert. |
| *planid* | Wenn eine JiffyBox aufgetaut wird, muss der Tarif angegeben werden, mit dem diese aufgetaut werden soll. Der Tarif muss in der Lage sein, die Daten aufzunehmen, sonst wird ein Fehler zurückgeliefert. Eine Liste aller gültigen Tarife können Sie unter der URL `/plans` abfragen. |
| *metadata* | Die zu einer JiffyBox hinterlegten Metadaten können geändert oder neue können hinzugefügt werden. Metadaten werden als Hash übergeben und genau so abgelegt, wie sie übergeben werden. Ist der Wert zu einem Schlüssel allerdings NULL, so wird der entsprechende Eintrag gelöscht. Alle Werte von nicht übergebenen Schlüsseln werden nicht verändert. Insgesamt dürfen die gespeicherten Metadaten bis zu 4 kb belegen. Bitte beachten Sie, dass der Platzbedarf sich auf den aus den Metadaten generierten JSON-String bezieht. |

## Beispiel

```
# curl -X PUT -d status=START -d 'metadata[createdBy]=Tester' \
      https://api.jiffybox.de/<Token>/v1.0/jiffyBoxes/12345
```

```
{
  "messages": [],
  "result": {
    "id": 12345,
    "name":"Test",
    "ips": { … }
    "status": "UPDATING",
    "created": 1234567890,
    "recoverymodeActive": false,
    "manualBackupRunning": false,
    "isBeingCopied": false,
    "running": false,
    "host": "vmhost-testsys-2-2-9-2",
```

```
    "metadata": {
      "createdBy": "Tester",
      "usedBy": ["Me", "You", "Everyone"],
      "freeForAll": false
    }
    "plan": { … }
    "activeProfile": { … }
  }
}
```

# Einen Tarifwechsel für eine JiffyBox durchführen

Verb: **PUT** URL: **/jiffyBoxes/<Box-ID>**

## Beschreibung

Um den Tarif einer JiffyBox zu ändern, ist ein PUT-Request nötig, der den neuen Tarif festlegt. Die zu modifizierenden Felder sind `status` und `planid`. Dieses Kommando wird komplett asynchron durchgeführt. Direkt nach dem Aufrufen ändert sich das Feld `status` auf `CHANGING PLAN`, um anzuzeigen, dass der Tarifwechsel durchgeführt wird. So lange `status` auf diesem Wert steht, akzeptiert die JiffyBox keine weiteren Kommandos. Bei Erfolg ändert sich `status` anschließend auf `READY`, um anzuzeigen, dass die JiffyBox jetzt bereit ist, weitere Kommandos entgegenzunehmen.

Das Feld `plan` wird erst nach erfolgreicher Durchführung des Tarifwechsels die Daten des neuen Tarifs reflektieren.

## Parameter

| | |
|---|---|
| **status** | Der einzutragende, neue Status. Es wird nur der Wert `PLAN` akzeptiert. |
| **planid** | Bei einem Tarifwechsel muss der Tarif angegeben werden, in den gewechselt werden soll. Der Tarif muss in der Lage sein, die vorhandenen Daten aufzunehmen, sonst wird ein Fehler zurückgeliefert. Eine Liste aller gültigen Tarife können Sie unter der URL `/plans` abfragen. |

## Beispiel

```
# curl -X PUT -d status=PLAN -d planid=20 \
       https://api.jiffybox.de/<Token>/v1.0/jiffyBoxes/12345
```

```
{
  "messages": [],
  "result": {
    "id": 12345,
    "status": "CHANGING PLAN",
    …
    "plan": {
      "id": 28,
      "name": "CloudLevel 1 SSD",
      "diskSizeInMB": 25600,
      "ramInMB": 2048,
      "pricePerHour": 0.025,
      "pricePerHourFrozen": 0.005,
      "cpus": 3
    },
  }
}
```

# Backup-Kommandos

## Alle Backups auflisten

Verb: **GET** URL: **/backups**

### Beschreibung

Um sich alle verfügbaren Backups aller JiffyBoxen anzeigen zu lassen, können Sie dieses Kommando nutzen. Es ist allerdings um ein Vielfaches langsamer, als das selektive Abfragen einer einzelnen JiffyBox, da die Backup-Server synchron nach der Verfügbarkeit der Backups gefragt werden. Wenn es möglich ist, sollten Sie deshalb immer den spezifischeren Aufruf nutzen.

## Beispiel

```
# curl https://api.jiffybox.de/<Token>/v1.0/backups
```


```
{
  "messages": [],
  "result": {
    "12345": {
      "daily": {
        "id": "1234567890abcdef1234567890abcdef",
        "created": 1234567890
      },
      "weekly": {
        "id": "234567890abcdef1234567890abcdef1",
        "created": 1234567890
      },
      "biweekly": {
        "id": "34567890abcdef1234567890abcdef12",
        "created": 1234567890
      }
    }
  }
}
```


 

# Backups zu einer JiffyBox auflisten

Verb: **GET** URL: **/backups/<Box-ID>**

## Beschreibung

Dieser Aufruf listet Details zu allen Backups der Jiffy-Box `Box-ID`. Da dieser Aufruf synchron ist, kann er unter Last länger dauern als normalerweise. Die Verfügbarkeit der Backups wird direkt auf den Backup-Storages geprüft. Hierbei können nur fertig erstellte Backups angezeigt werden.

Die Rückgabewerte `day` und `time` geben an, ob und wann jeweils automatisch das tägliche Backup durchgeführt wird. Wenn sie keine Werte zurückliefern, sind automatische Backups deaktiviert. `day` ist ein Wert zwischen 0 (Sonntag) und 6 (Samstag). `time` ist entweder 0 (zwischen 0:00 und 4:00 Uhr morgens), 1 (zwischen 4:00 und 6:00 Uhr morgens) oder 6 (zwischen 22:00 und 0:00 Uhr).

## Beispiel

```
# curl https://api.jiffybox.de/<Token>/v1.0/backups/12345
```

```
{
  "messages": [],
  "result": {
    "daily": {
      "id": "1234567890abcdef1234567890abcdef",
      "created": 1234567890
    },
    "weekly": {
      "id": "234567890abcdef1234567890abcdef1",
      "created": 1234567890
    },
    "biweekly": {
      "id": "34567890abcdef1234567890abcdef12",
      "created": 1234567890
    },
    "day": 1,
    "time": 1
  }
}
```

# Backups für eine JiffyBox aktivieren

Verb: **POST** URL: **/backups/<Box-ID>**

## Beschreibung

Über dieses Kommando können Sie einstellen, wenn für eine JiffyBox regelmäßig Backups erzeugt werden sollen.

## Parameter

| | |
|---|---|
| **dayid** | Der Wochentag, an dem jeweils ein Backup durchgeführt werden soll. Gültige Werte sind 0 (sonntags) bis 6 (samstags). |
| **timeid** | Der Zeitrahmen, wann das Backup jeweils erstellt werden soll. Mögliche Werte sind:<br>**1** Zwischen 0:00 und 4:00 Uhr morgens<br>**2** Zwischen 4:00 und 6:00 Uhr morgens<br>**6** Zwischen 22:00 und 0:00 Uhr |

## Beispiel

```
# curl -d dayid=1 -d timeid=2 \
       https://api.jiffybox.de/<Token>/v1.0/backups/12345
```

```
{
  "messages": [],
  "result": {
    "dayid": 1,
    "timeid": 2
  }
}
```

# Die Backup-Konfiguration ändern

Verb: **PUT** URL: **/backups/<Box-ID>**

## Beschreibung

Mit diesem Kommando können Sie eine bereits existierende Backup-Konfiguration ändern. Zum Neuanlegen bzw. Aktivieren der automatischen Backups nutzen Sie bitte das `POST`-Verb.

## Parameter

Die Parameter sind die gleichen wie bei der `POST`-Version, mit der Ausnahme, dass alle Parameter optional sind und nur die übergebenen Werte geändert werden.

## Beispiel

```
# curl -X PUT -d dayid=2 \
       https://api.jiffybox.de/<Token>/v1.0/backups/12345
```

```
{
  "messages": [],
  "result": {
    "dayid": 2,
    "timeid": 2
  }
}
```

# Backups für eine JiffyBox deaktivieren

Verb: **DELETE** URL: **/backups/<Box-ID>**

## Beschreibung

Standardmäßig werden jeden Tag automatisch Backups jeder JiffyBox erzeugt. Sollte dies einmal unerwünscht sein (z.B. wegen zu hoher Last), lassen sich diese automatischen Backups wieder deaktivieren. Bitte beachten Sie, dass Sie beim Deaktivieren der automatischen Backups alle bisher erstellen automatischen Backups verlieren. Auf die manuell erstellten Backups sowie die Möglichkeit selbige zu erstellen, hat dieser Aufruf keine Auswirkungen.

## Beispiel

```
# curl -X DELETE https://api.jiffybox.de/<Token>/v1.0/backups/12345
```

```
{
  "messages": [],
  "result":true
}
```

# Manuelles Backup erstellen

Verb: **POST** URL: **/backups/<Box-ID>/manual**

## Beschreibung

Jede JiffyBox kann genau ein manuelles Backup halten. Diese Methode kann genutzt werden, um ein manuelles Backup anzufordern. Der Vorgang selbst ist komplett asynchron. Ein Erfolg (Rückgabewert `TRUE`) bedeutet also nur, dass das manuelle Backup erfolgreich beantragt wurde, nicht dass es gestartet oder gar abgeschlossen ist. Ob ein manuelles Backup läuft, können Sie über `/jiffyBoxes` auslesen. Dort existiert das Flag `manualBackupRunning`. Diese Flag ist `TRUE`, falls das Backup gerade läuft. Sobald das Backup abgeschlossen ist, erscheint es in der Backup-Übersicht.

## Beispiel

```
# curl -d "" htts://api.jiffybox.de/<Token>/v1.0/backups/12345/manual
```

```
{
  "messages": [],
  "result":true
}
```

# Ein Backup löschen

Verb: **DELETE** URL: **/backups/<Box-ID>/<Typ>/<Backup-ID>**

## Beschreibung

Mit diesem Kommando ist es möglich, ein Backup anhand seines Typs (`manual`, `daily`, `weekly`, `biweekly`) und seiner eindeutigen ID zu löschen. Typ und ID können Sie über die entsprechenden `GET`-Kommandos auslesen.

## Beispiel

```
# curl -X DELETE \
       https://api.jiffybox.de/<Token>/v1.0/backups/12345/daily/12345ACDEF
```

```
{
  "messages": [],
  "result": true
}
```

# Ein Backup wiederherstellen

Verb: **POST** URL: **/backups/<Box-ID>/<Typ>/<Backup-ID>**

## Beschreibung

Mit diesem Kommando können Sie den Inhalt der JiffyBox `Box-ID` durch das Backup `Backup-ID` ersetzen. Alle derzeit vorhandenen Daten gehen verloren und die Konfiguration der JiffyBox (Profile, Festplatten) wird auf den Zustand zum Zeitpunkt des Backups versetzt. Eine Liste aller gültigen Backup-Typen und -IDs erhalten Sie per `GET`-Request auf `/backups/<Box-ID>`.

## Beispiel

```
# curl -d '' \
       https://api.jiffybox.de/<Token>/v1.0/backups/12345/daily/12345ACDEF
```

```
{
  "messages": [],
  "result": true
}
```

# Tarif-Kommandos

## Alle Tarife auflisten

Verb: **GET** URL: **/plans**

### Beschreibung

Liefert eine Liste aller bestellbaren Tarife. Wann immer ein Tarif referenziert wird, können Sie entweder `id` oder `name` übergeben. Die Schlüsselwerte unterhalb von `result` entsprechen der `id`.

### Beispiel

```
# curl https://api.jiffybox.de/<Token>/v1.0/plans
```


```
{
  "messages": [],
  "result": {
    "20": {
  "id":20,
  "name":"CloudLevel 1",
  "diskSizeInMB":76800,
  "ramInMB":2048,
  "pricePerHour":0.02,
  "pricePerHourFrozen":0.005,
  "cpus":3
    },
    "21": {
  "id":21,
  "name":"CloudLevel 2",
  "diskSizeInMB":153600,
  "ramInMB":4096,
  "pricePerHour":0.04,
  "pricePerHourFrozen":0.01,
  "cpus":4
    },
    "22": {
  "id":22,
  "name":"CloudLevel 3",
  "diskSizeInMB":307200,
  "ramInMB":8192,
  "pricePerHour":0.07,
  "pricePerHourFrozen":0.02,
  "cpus":6},
    },
    "23": {
  "id":23,
  "name":"CloudLevel 4",
  "diskSizeInMB":409600,
  "ramInMB":16384,
  "pricePerHour":0.13,
  "pricePerHourFrozen":0.03,
  "cpus":6
    }
  }
}
```


 

## Details zu einem Tarif ansehen

Verb: **GET** URL: **/plans/<id oder name>**

### Beschreibung

Liefert die Details zu einem bestimmten Tarif. Der Tarif kann entweder über seine ID oder den Namen angesprochen werden.

### Beispiel

```
# curl https://api.jiffybox.de/<Token>/v1.0/plans/11
# curl https://api.jiffybox.de/<Token>/v1.0/plans/CloudLevel%202
```


```
{
  "messages": [],
  "result": {
    "id": 21,
    "name": "CloudLevel 2",
    "diskSizeInMB": 153600,
    "ramInMB": 4096,
    "pricePerHour": 0.04,
    "pricePerHourFrozen": 0.01
    "cpus": 4
  }
}
```

# Linux-Distributions-Aufrufe

## Alle Linux-Distributionen auflisten

Verb: GET URL: /distributions

### Beschreibung

Listet alle Linux-Distributionen auf, die zur Initialisierung von Festplatten und JiffyBoxen verfügbar sind. Immer wenn diese Linux-Distributionen von anderen Kommandos referenziert werden, muss der Schlüsselwert des Hashes, z.B. centos_5_4_32bit, genommen werden.

### Beispiel

```
# curl https://api.jiffybox.de/<Token>/v1.0/distributions
```


```
{
  "messages": [],
  "result": {
    "centos_5_4_32bit": {
      "minDiskSizeMB": 1024,
      "name": "CentOS 5.4",
      "rootdiskMode": "ro",
      "defaultKernel": "xen-current"
    },
    "centos_5_4_64bit": {
      "minDiskSizeMB": 1024,
      "name": "CentOS 5.4 64-Bit",
      "rootdiskMode": "ro",
      "defaultKernel": "xen-current-x86_64"
    }
  }
}
```


 

# Details zu einer Linux-Distribution erfragen

Verb: **GET** URL: **/distributions/<id>**

## Beschreibung

Mit diesem Aufruf erhalten Sie Details zu einer speziellen Linux-Distribution. `id` ist der Hash-Schlüsselwert, der beim Auslesen der Gesamtliste geliefert wird.

## Beispiel

```
# curl https://api.jiffybox.de/<Token>/v1.0/distributions/centos_5_4_64bit
```

```
{
  "messages": [],
  "result": {
    "minDiskSizeMB": 1024,
    "name": "CentOS 5.4 64-Bit",
    "rootdiskMode": "ro",
    "defaultKernel": "xen-current-x86_64"
  }
}
```

# IP-Kommandos

## Alle IP-Adressen auflisten

Verb: **GET** URL: **/ips**

### Beschreibung

Um sich alle IP-Adressen aller JiffyBoxen anzeigen zu lassen, können Sie dieses Kommando nutzen.

Der Rückgabewert `isSubnet` gibt an, ob es sich bei der IP um eine einzelne Adresse oder ein Subnetz handelt. Ein Subnetz kann über das Control-Panel erstellt werden und zwar ausschließlich für IPv6-Netze.

Der Wert `reverseLookup` gibt den Reverse-DNS-Namen der IP-Adresse an. Für Subnetze lässt sich der DNS-Lookup optional auf einen fremden Nameserver delegieren.

Der Rückgabewert `type` gibt an, ob es sich um eine öffentliche oder private (nicht routbare) IP-Adresse handelt.

Der Wert `floating` sagt aus, ob diese Adresse zwischen JiffyBoxen verschoben werden kann, um damit beispielsweise einen Failover-Mechanismus zu implementieren. Das Verschieben von JiffyBoxen ist nur für Ipv4-Adressen möglich, die über das Control-Panel zusätzlich bestellt wurden.

### Beispiel

```
# curl https://api.jiffybox.de/<Token>/v1.0/ips
```


```
{
  "messages":[],
  "result": {
    "12345": {
        "8376": {
          "id":8376,
          "ip":"188.1.2.3",
          "ipVersion":4,
          "isSubnet":false,
          "reverseLookup":"blah.blubb.de",
          "type":"public",
          "floating":"false"
        },
        "8377": {
          "id":8377,
          "ip":"10.1.2.3",
          "ipVersion":4,
          "isSubnet":false,
          "reverseLookup":"j4192.servers.jiffybox.net",
          "type":"private",
          "floating":"false"
        },
        "8463": {
          "id":8463,
          "ip":"2a00:1234:1:234::1",
          "ipVersion":6,
          "isSubnet":false,
          "reverseLookup":"test2.de",
```


 

```
          "type":"public",
          "floating":"false"
        },
        "8467": {
          "id":8467,
          "ip":"2a00:1234:1:211::/56",
          "ipVersion":6,
          "isSubnet":true,
          "reverseLookup": {
              "delegatedTo":["ns1.de","ns2.de"]
          },
          "type":"public",
          "floating":"false"
        },
        "8465": {
          "id":8465,
          "ip":"188.1.4.241",
          "ipVersion":4,
          "isSubnet":false,
          "reverseLookup":"ip-188.1.4.241.servers.jiffybox.net",
          "type":"public",
          "floating":"true"
        },
    }
  }
}
```

# IP-Adressen zu einer JiffyBox auflisten

Verb: **GET** URL: **/ips/<Box-ID>**

## Beschreibung

Listet alle IP-Adressen auf die der JiffyBox Box-ID zugeordnet sind.

## Beispiel

```
# curl https://api.jiffybox.de/<Token>/v1.0/ips/12345
```


```
{
  "messages":[],
  "result": {
      "8376": {
        "id":8376,
        "ip":"188.1.2.3",
        "ipVersion":4,
        "isSubnet":false,
        "reverseLookup":"blah.blubb.de",
        "type":"public",
        "floating":"false"
      },
      "8377": {
        "id":8377,
        "ip":"10.1.2.3",
        "ipVersion":4,
        "isSubnet":false,
        "reverseLookup":"j4192.servers.jiffybox.net",
        "type":"private",
        "floating":"false"
      },
      "8463": {
        "id":8463,
        "ip":"2a00:1234:1:234::1",
        "ipVersion":6,
        "isSubnet":false,
        "reverseLookup":"test2.de",
        "type":"public",
        "floating":"false"
      },
      "8467": {
        "id":8467,
        "ip":"2a00:1234:1:211::/56",
        "ipVersion":6,
        "isSubnet":true,
        "reverseLookup": {
            "delegatedTo":["ns1.de","ns2.de"]
        },
        "type":"public",
        "floating":"false"
      },
      "8465": {
        "id":8465,
        "ip":"188.1.4.241",
        "ipVersion":4,
        "isSubnet":false,
        "reverseLookup":"ip-188.1.4.241.servers.jiffybox.net",
        "type":"public",
```

```
          "floating":"true"
        },
      }
    }
}
```

# IP-Adresse einer anderen JiffyBox zuweisen

Verb: **PUT** URL: **/ips/<Box-ID>/<IP-ID>/move**

## Beschreibung

Über dieses Kommando können Sie IPv4-Adressen zwischen JiffyBoxen verschieben. Dies funktioniert ausschließlich für über das Control-Panel bestellte zusätzliche IP-Adressen. Das Kommando beeinflusst ausschließlich das Routing im JiffyBox-Netzwerk. Um die IP auf der Ziel-Jiffybox nutzen zu können muss Sie im dortigen Linux normal konfiguriert werden.

## Parameter

| | |
|---|---|
| **targetid** | Die Ziel-JiffyBox, der die IP-Adresse zugewiesen werden soll. |

## Beispiel

```
# curl -d targetid=4321 -X PUT \
https://api.jiffybox.de/<Token>/v1.0/ips/12345/8465/move
```

```
{
"messages": [],
"result": true
}
```

# Monitoring-Kommandos

Monitoring-Checks können folgende Statuswerte haben:

```
STATUS_ERROR_RETRY
STATUS_ERROR
STATUS_OFFLINE
STATUS_READY
STATUS_CREATING
STATUS_CREATED
STATUS_UPDATING
STATUS_UPDATED
STATUS_DEACTIVATING
STATUS_DEACTIVATED
STATUS_DELETING
STATUS_DELETED
```

Die Differenzierung `STATUS_UPDATING` / `STATUS_UPDATED` liegt in der asynchronen Funktionsweise des Monitorings begründet. `STATUS_UPDATED` bedeutet, dass der Monitoring-Check erfolgreich an den Monitoring-Server übermittelt wurde, die Änderungen allerdings erst nach dem nächsten Neuladen der Konfiguration auf dem Server wirksam werden. Sobald die Monitoring-Konfiguration neu geladen wurde, wechselt `status` auf den Wert `STATUS_READY`. Ein ähnliches Verhalten gilt auch für alle anderen paarweise angelegten Statuswerte.

# Alle Monitoring-Checks auflisten

Verb: **GET** URL: **/monitoring**

## Beschreibung

Mit diesem Aufruf erhalten Sie eine Übersicht aller Monitoring-Checks.

## Beispiel

```
# curl https://api.jiffybox.de/<token>/v1.0/monitoring
```


```
{
  "messages":[],
  "result":{
    "911":{
      "id":1234,
      "name":"test",
      "jiffyBox":4849,
      "ip":"188.93.14.165",
      "checkType":"http",
      "checkInterval":300,
      "reminderInterval":3600,
      "retryTolerance":3,
      "status":"STATUS_READY",
      "port":80,
      "path":"\/index\.php",
      "domainname":"example.com",
      "username":"",
      "password":"",
      "contactgroups":[{"id":123,"name":"TestGruppe"}]
    }
  }
}
```


 

# Details zu einem Monitoring-Check abfragen

Verb: **GET** URL: **/monitoring/<Check-ID>**

## Beschreibung

Liefert Details zu einem bestimmen Monitoring-Check. Der Rückgabewert entspricht dem eines einzelnen Monitoring-Checks in `/monitoring`.

## Beispiel

```
# curl https://api.jiffybox.de/<token>/v1.0/monitoring/1234
```


```
{
  "messages": [],
  "result": {
    "id":1234,
    "name":"test",
    "jiffyBox":4849,
    "ip":"188.93.14.211",
    "checkType":"http",
    "checkInterval":300,
    "reminderInterval":3600,
    "retryTolerance":3,
    "status":"STATUS_READY",
    "port":80,
    "path":"\/index\.php",
    "domainname":"example.com",
    "username":"",
    "password":"",
    "contactgroups":[{"id":123,"name":"TestGruppe"}]
 }
}
```


 

# Einen Monitoring-Check löschen

Verb: **DELETE** URL: **/monitoring/<Check-ID>**

## Beschreibung

Mit diesem Kommando können Sie Monitoring-Checks löschen. Das Kommando ist asynchron. Bei Erfolg wird `true` geliefert, bei einem Fehler `false`. Ein Erfolg bedeutet lediglich, dass das Löschen erfolgreich beantragt wurde. Das eigentliche Löschen findet asynchron statt. Der Status der Löschung kann durch Auslesen der Monitoring-Details erfragt werden. Sobald die Meldung kommt, dass der Monitoring-Check nicht mehr existiert, ist der Löschvorgang abgeschlossen. Während des Löschvorgangs ist der Status des Monitoring-Checks `DELETING` / `DELETED`. Sollte der Status wieder auf `READY` springen, war der Löschvorgang nicht erfolgreich.

## Beispiel

```
# curl -X DELETE https://api.jiffybox.de/<token>/v1.0/monitoring/1234
```

```
{
  "messages": [{"type":"success","message":"Der Monitoring-Check Test
(M1234) wird geloescht"}],
  "result": true
}
```

# Einen Monitoring-Check neu erstellen

Verb: **POST** URL: **/monitoring**

## Beschreibung

Es gibt zwei Möglichkeiten, einen Monitoring-Check zu erstellen:

1. Den Monitoring-Check komplett neu erstellen
2. Ihn als Kopie eines vorhandenen Monitoring-Checks erstellen

Das Erstellen eines Monitoring-Checks ist ein asynchroner Vorgang. Ein Erfolg als Rückgabe bedeutet nur, dass das Erstellen gestartet wurde. Über das Ergebnis kann nur ein periodisches Abfragen der Monitoring-Details via `GET`-Request auf `/monitoring/<id>` Auskunft geben. `id` ist in den Rückgabewerten der Erstellung enthalten. Während des Erstellens ist der Status des Monitoring-Checks auf `CREATING`. Wenn der Monitoring-Check fertig eingerichtet ist, wechselt der Status bei Erfolg auf `READY`. Bei Misserfolg wird der Vorgang einmal automatisch wiederholt. Wenn es dennoch nicht gelingt den Monitoring-Check zu erstellen, wird der Status auf `ERROR` gesetzt.
Alle Parameter, die Zeiten bestimmen, müssen in **Sekunden** angegeben werden. Werden die optionalen Parameter nicht angegeben, wird die Voreinstellung verwendet.

## Parameter

| | |
|---|---|
| **name** | Der Name des Monitoring-Checks. Erlaubt sind bis zu 30 Zeichen. Erlaubte Zeichen sind: `a-zA-Z0-9üöäÜÖÄß_()=!\*@.-` und Leerzeichen. |
| **ip** | IP-Adresse, an die der Monitoring-Check gebunden ist. |
| **checkType** | Typ des Monitoring-Checks. Zur Auswahl stehen: ping, portTcp, protUdp, http, https, smtp, pop3, imap, ftp, ssh, mysql, dns. |
| *checkInterval* | Das Checkintervall bestimmt, in welchen zeitlichen Abständen der Monitoring-Check geprüft wird. Voreinstellung : 5 Minuten (300 Sekunden). |
| *reminderInterval* | Das Erinnerungsintervall bestimmt, in welchen zeitlichen Abständen E-Mails versendet werden sollen. Voreinstellung: 1 Stunde (3600 Sekunden). |
| *retryTolerance* | Die Benachrichtigungstoleranz bestimmt, wie oft ein Monitoring-Check wiederholt kritisch werden muss, bis eine Benachrichtigung über das Ergebnis des Checks ausgelöst wird. Voreinstellung: 3. |
| *timeout* | Der Timeout-Wert bestimmt, nach welcher Zeit ein erfolgloser Check-Versuch abgebrochen wird. Voreinstellung: 30. |
| *active* | Monitoring-Checks können sowohl aktiv (asynchroner Vorgang) erstellt und nachdem die Monitoring-Konfiguration neugeladen wurde, überprüft werden, als auch inaktiv nur abgespeichert werden. |

| | |
|---|---|
| *contactgroups* | Eine oder mehrere Kontakt-Gruppen-IDs, an die E-Mail-Benachrichtigungen geschickt werden. Wird dieser Parameter nicht geschickt, sind die Ergebnisse des Monitorings nur im Control-Panel einsehbar. |

## Zusätzliche Parameter

Je nach Check-Typ existieren zusätzlich die folgenden Parameter:

| Parameter | Pflicht für folgende Check-Typen | *Optional für folgende Check-Typen* |
|---|---|---|
| port | alle | – |
| username | smtp, pop3, imap, mysql | http, https, ftp |
| password | smtp, pop3, imap, mysql | http, https, ftp |
| ssl | – | pop3, imap |
| path | http, https | – |
| domainname | http, https | – |
| sendMsg | portUdp | – |
| expectMsg | portUdp | – |
| dnsType | dns | – |
| zone | dns | – |
| expectedAnswer | dns | – |

## Beispiele

```
# curl -d name=Test -d ip="188.93.14.211" -d checkType="http" -d port=80 \
-d ip="188.93.14.165" -d path="/index.php" -d domainname="example.com" \
      https://api.jiffybox.de/<Token>/v1.0/monitoring
```


```
{
  "messages": [],
  "result": {
    "id":1234,
    "name":"Test",
    "jiffyBox":4849,
    "ip":"188.93.14.211",
    "checkType":"http",
    "checkInterval":300,
    "reminderInterval":3600,
    "retryTolerance":3,
    "status":"STATUS_CREATING",
    "port":80,
    "path":"\/index.php",
    "domainname":"example.com",
    "contactgroups":[]
```


 

```
    }
}
```

# Einen Monitoring-Check duplizieren

Verb: **POST** URL: **/monitoring/<Check-ID>**

## Beschreibung

Duplizieren von Monitoring-Checks benötigt die `id` des Monitoring-Checks als `GET`-Parameter. Um einen Monitoring-Check zu duplizieren, muss zumindestens ein Parameter geändert werden - alle sonstigen Einstellungen werden vom Original-Monitoring-Check übernommen. Bei den Parametern gelten die Hinweise aus dem Kapitel zum Erstellen eines neuen Monitoring-Checks.

## Beispiel

```
curl -d name="Kopie von Test" -d ip="188.93.14.212"
https://api.jiffybox.de/<Token>/v1.0/monitoring/1234
```


```
{
  "messages":[],
  "result":  {
    "id":1235,
    "name":"Kopie von Test",
    "jiffyBox":4849,
    "ip":"188.93.14.212",
    "checkType":"http",
    "checkInterval":300,
    "reminderInterval":3600,
    "retryTolerance":3,
    "status":"STATUS_CREATING",
    "port":80,
    "path":"\/index.php",
    "domainname":"example.com",
    "contactgroups":[]
  }
}
```


 

# Den Status eines Monitoring-Checks abrufen

Verb: GET URL: **/monitoring/<Check-ID>/status**

## Beschreibung

Liefert den aktuellen Status eines einzelnen aktiven Monitoring-Checks.

## Beispiel

```
# curl https://api.jiffybox.de/<token>/v1.0/monitoring/1234/status
```

```
{
  "messages": [],
  "result": {
    "1234": {
      "code": 0,
      "response": "OK - 123.45.67.89: rta 0.313ms, lost 0%"
    }
  }
}
```

# Den Status mehrerer Monitoring-Checks abrufen

Verb: GET URL: **/monitoring/<IPv4-Adresse>/status**

## Beschreibung

Mit diesem Aufruf erhalten Sie den aktuellen Status aller für eine IP-Adresse definierten und aktiven Monitoring-Checks.

## Beispiel

```
# curl https://api.jiffybox.de/<token>/v1.0/monitoring/123.45.67.89/status
```

```
{
  "messages": [],
  "result": {
    "1234": {
      "code": 0,
      "response": "OK - 123.45.67.89: rta 0.313ms, lost 0%"
    },
    "5678": {
      "code": 0,
      "response": "HTTP OK: Status line output matched &quot;200&quot; -
3827 bytes in 0.003 second response time"
    }
  }
}
```

# Kontaktgruppen-Kommandos

Kontaktgruppen sind Gruppen von E-Mail-Adressen, die vom JiffyBox-Monitoring im Falle eines Alarms benachrichtigt werden. Pro JiffyBox-Account kann es bis zu 5 frei definierte Kontaktgruppen geben; jede dieser Kontaktgruppen kann bis zu 10 E-Mail-Adressen enthalten.

Sobald Sie für eine JiffyBox das Monitoring erstmals aktivieren, wird eine vordefinierte Kontaktgruppe „Stammdaten-E-Mail-Adresse" erstellt, die als einzigen Empfänger die E-Mail-Adresse enthält, die in den Stammdaten gespeichert ist. Diese Kontaktgruppe kann über die API nicht verändert oder gelöscht werden.

Kontaktgruppen können folgende Statuswerte haben:

```
STATUS_ERROR_RETRY
STATUS_ERROR
STATUS_OFFLINE
STATUS_READY
STATUS_UPDATING
STATUS_UPDATED
STATUS_DELETING
STATUS_DELETED
```

Die Differenzierung `STATUS_UPDATING` / `STATUS_UPDATED` liegt in der asynchronen Funktionsweise des Monitorings begründet. `STATUS_UPDATED` bedeutet in diesem Fall, dass die Kontaktgruppe erfolgreich an den Monitoring-Server übermittelt wurde, die Änderungen allerdings erst nach dem nächsten Neuladen der Konfiguration auf dem Server wirksam werden. Sobald die Monitoring-Konfiguration neu geladen wurde, wechselt `status` auf den Wert `STATUS_READY`.

# Alle Kontaktgruppen auflisten

Verb: **GET** URL: **/contactGroups**

## Beschreibung

Mit diesem Aufruf erhalten Sie eine Übersicht aller Kontaktgruppen.

## Beispiel

```
# curl https://api.jiffybox.de/<token>/v1.0/contactGroups
```


```
{
  "messages": [],
  "result": {
    "122": {
      "id": 122,
      "name": "Stammdaten-E-Mail-Adresse",
      "contacts": [
        "m.mustermann@example.com"
      ],
      "status": "STATUS_READY"
    },
    "123": {
      "id": 123,
      "name": "TestGruppe",
      "contacts": [
        "m.mustermann@example.com"
      ],
      "status": "STATUS_READY"
    }
  }
}
```

# Details zu einer Kontaktgruppe abfragen

Verb: **GET** URL: **/contactGroups/<Group-ID>**

## Beschreibung

Liefert Details zu einer bestimmen Kontaktgruppe. Der Rückgabewert entspricht dem einer einzelnen Kontaktgruppe in `/contactGroups`.

## Beispiel

```
# curl https://api.jiffybox.de/<token>/v1.0/contactGroups/123
```

```
{
  "messages": [],
  "result": {
    "id": 123,
    "name": "TestGruppe",
    "contacts": [
      "m.mustermann@example.com"
    ],
    "status": "STATUS_READY"
  }
}
```

# Eine Kontaktgruppe löschen

Verb: **DELETE** URL: **/contactGroup/<Group-ID>**

## Beschreibung

Mit diesem Kommando können Sie eine Kontaktgruppe löschen. Das Kommando ist asynchron. Bei Erfolg wird `true` geliefert, bei einem Fehler `false`. Ein Erfolg bedeutet lediglich, dass das Löschen erfolgreich gestartet wurde. Das eigentliche Löschen findet asynchron statt. Der Status der Löschung kann durch Auslesen der Kontaktgruppe erfragt werden. Sobald die Meldung kommt, dass die Kontaktgruppe nicht existiert, ist der Löschvorgang abgeschlossen. Während des Löschvorgangs ist der Status der Kontaktgruppe `STATUS_DELETING` / `STATUS_DELETED`. Sollte dieser wieder auf `STATUS_READY` springen, war der Löschvorgang nicht erfolgreich.

## Beispiel

```
# curl -X DELETE https://api.jiffybox.de/<token>/v1.0/contactGroups/123
```

```
{
  "messages": [],
  "result": true
}
```

# Eine Kontaktgruppe neu erstellen

Verb: **POST** URL: **/contactGroup**

## Beschreibung

Es gibt zwei Möglichkeiten, eine Kontaktgruppe zu erstellen:

1. Die Kontaktgruppe komplett neu erstellen
2. Eine vorhandene Kontaktgruppe kopieren

Das Erstellen einer Kontaktgruppe ist ein asynchroner Vorgang. Ein Erfolg als Rückgabe bedeutet nur, dass das Erstellen gestartet wurde. Über das Ergebnis kann nur ein periodisches Abfragen der Details via `GET`-Request auf `/contactGroup/<id>` Auskunft geben. `id` ist in den Rückgabewerten der Erstellung enthalten. Während des Erstellens steht der Inhalt des Felds `status` auf `STATUS_CREATING`. Wenn die Kontaktgruppe fertig eingerichtet ist, wechselt der Status bei Erfolg auf `STATUS_READY`. Bei Misserfolg wird der Vorgang einmal automatisch wiederholt. Wenn es dennoch nicht gelingt, die Kontakt-Gruppe zu erstellen, wird der Status auf `STATUS_ERROR` gesetzt.

## Parameter

| | |
|---|---|
| **name** | Der Name der Kontakt-Gruppe. Erlaubt sind bis zu 30 Zeichen. Erlaubte Zeichen sind: `a-zA-Z0-9üöäÜÖÄß_()=!\*@.-` und Leerzeichen. |
| **contacts** | Ein Array aus maximal 10 E-Mail-Adressen |

## Beispiele

```
curl -X POST \
  -d name="TestGruppe" \
  -d contacts=m.mustermann@df.eu \
  -d contacts=f.musterfrau@df.eu \
  https://api.jiffybox.de/<Token>/v1.0/contactGroups
```

```
{
  "messages": [],
  "result": {
    "id": 1234,
    "name": "TestGruppe",
    "contacts": [
      "m.mustermann@example.com"
      "f.musterfrau@example.com"
    ],
    "status":"STATUS_READY"
  }
}
```

# Eine Kontaktgruppe modifizieren

Verb: **PUT** URL: /**contactGroup**

## Beschreibung

Mit diesem Kommando können Sie Änderungen an einer bereits bestehenden Kontaktgruppe vornehmen. Hiervon ausgenommen ist die Kontaktgruppe „Stammdaten-E-Mail-Adresse", die nicht verändert werden kann. Das Verhalten des Feldes `status` ist so, wie bereits beim Erstellen einer Kontaktgruppe aufgeführt.

## Parameter

| | |
|---|---|
| **name** | Der Name der Kontakt-Gruppe. Erlaubt sind bis zu 30 Zeichen. Erlaubte Zeichen sind: `a-zA-Z0-9üöäÜÖÄß_()=!\*@.-` und Leerzeichen. |
| **contacts** | Ein Array aus maximal 10 E-Mail-Adressen |

## Beispiele

```
curl -X PUT \
  -d name="Neuer Name der TestGruppe" \
  https://api.jiffybox.de/<Token>/v1.0/contactGroups
```

```
{
  "messages": [],
  "result": {
    "id": 1234,
    "name": "Neuer Name der TestGruppe",
    "contacts": [
      "m.mustermann@example.com"
      "f.musterfrau@example.com"
    ],
    "status": "STATUS_UPDATING"
  }
}
```

# Eine Kontaktgruppe duplizieren

Verb: **POST** URL: /contactGroup/**<Group-ID>**

## Beschreibung

Das Duplizieren von Kontakt-Gruppen benötigt die `id` der Ursprungs-Kontakt-Gruppe als Parameter. Um eine Kontakt-Gruppe zu duplizieren, muss zumindest ein Parameter geändert werden, üblicherweise der Parameter `name`.

## Beispiel

```
# curl -X POST
  -d name="Kopie von TestGruppe" \
  https://api.jiffybox.de/<Token>/v1.0/contactGroups/1234
```

```
{
  "messages":[],
  "result": {
    "id": 1235,
    "name": "Kopie von TestGruppe",
    "contacts": [
      "m.mustermann@example.com"
      "f.musterfrau@example.com"
    ],
    "status": "STATUS_UPDATING"
  }
}
```

# Dokumentations-Kommandos

## Eine Liste aller verfügbaren Dokumentationsmodule erfragen

Verb: GET URL: /doc

### Beschreibung

Liefert eine Liste aller Module inklusive einer Kurzbeschreibung derselben. Über den Namen kann dann die Dokumentation im Detail erfragt werden.

### Beispiel

```
# curl https://api.jiffybox.de/<Token>/v1.0/doc
```

```
{
  "messages": [],
  "result": {
    "doc": "Dokumentationsmodul",
    "distributions": "Modul zum Auflisten von installierbaren Linux-
                      Distributionen",
    "jiffyBoxes": "Modul zum Auflisten und Manipulieren von
                   JiffyBoxen",
    "backups": "Modul zum Auflisten und Beantragen von Backups",
    "plans": "Modul zum Auflisten von JiffyBox-Tarifen"
  }
}
```

# Die Dokumentation eines Moduls abfragen

Verb: **GET** URL: **/doc/<modul>**

## Beschreibung

Liefert eine Kurzdokumentation eines Moduls. Eine Liste der Module gibt es unter `/doc`.

## Beispiel

```
# curl https://api.jiffybox.de/<Token>/v1.0/doc/doc
```


```
{
  "messages": [],
  "result": {
    "description": "Dokumentationsmodul",
    "\/": {
      "GET": {
        "description": "Hiermit bekommen Sie eine \u00dcbersicht aller
                        verf\u00fcgbaren Module mit einer
                        Kurzbeschreibung, was diese machen.",
        "parameters": {
          "must": {},
          "may": {}
        }
      }
    },
    "\/<module>": {
      "GET": {
        "description": "Hiermit lesen Sie die Kurzdokumentation eines
                        Moduls aus. Eine Liste aller Module gibt es
                        unter \/.",
        "parameters": {
          "must": {},
          "may": {}
        }
      }
    }
  }
}
```


 

# Änderungen

| Datum | API | Änderung |
|---|---|---|
| 2014-04-04 | 1.0 | Neues Kapitel „Einen Tarifwechsel für eine JiffyBox durchführen“ |
| 2013-12-04 | 1.0 | Kapitel „IP-Adresse einer anderen JiffyBox zuweisen“: Korrektur des Verbs. Hierbei handelt es sich, wie aus dem Beispiel korrekt hervorging, um einen PUT-Request. |
| 2013-12-02 | 1.0 | Neue Kapitel „Den Status eines Monitoring-Checks abrufen“ und „Den Status mehrerer Monitoring-Checks abrufen“ |
| 2013-11-11 | 1.0 | Neue Kapitel „Monitoring” und „Kontaktgruppen” eingefügt. |
| 2013-09-16 | 1.0 | Die URLs zu den Produktseiten von JiffyBox sowie die URL zum Forum (Kapitel „Einführung“) haben sich geändert und wurden aktualisiert. |
| 2013-07-08 | 1.0 | Status-Änderung für PUT-Request auf JiffyBox erklärt. |
| 2013-05-06 | 1.0 | Durch die Einführung neuer Leistungsstufen gibt es neue Plan-IDs. Die Plan-IDs der alten Leistungsstufen werden in der API am Mittwoch, den 15.05.2013, deaktiviert. |
| 2013-02-15 | 1.0 | Neues Kapitel „Eine JiffyBox duplizieren“ eingefügt. |
| 2012-01-31 | 1.0 | Durch die Einführung von IPv6 für JiffyBox werden im Modul JiffyBoxen nicht nur IPv4-Adressen sondern - bei entsprechender Konfiguration des Anwenders - auch IPv6-Adressen zurückgegeben.<br>Neues Modul „IPs“ eingefügt. |
| 2011-08-11 | 1.0 | Neues Feld im Rückgabewert für plans: „cpus“. Gibt die Anzahl der CPU-Kerne an. |