KEYWEB AKTIENGESELLSCHAFT

Serverstatus - API

zum Zugriff auf den Server-Statusbereich

Version 1.0.1.6
07/2019

Inhalt

I. Einleitung

Die ServerStatus-API stellt Ihnen und Ihren Kunden erweiterte Zugriffmöglichkeiten auf die Eigenschaften Ihrer gemieteten Webserver transparent zur Verfügung, ohne daß eine passwortgeschützte Weboberfläche genutzt werden muß.

Der Zugriff auf die ServerStatus-API erfolgt wahlweise über:

eine maskierbare IP-Adresse:

http://95.169.160.11/
https://95.169.160.11/

oder über die URL:

http://xml.status.keyweb.de/
https://xml.status.keyweb.de/

Der Zugriff über die IP-Adressen hat den Vorteil, das Sie diese Adressen über einen Ihrer Nameserver auflösen lassen können (z.B. xyz.meinepraesenz.de => 95.169.160.11) und somit ein transparenter Zugriff erfolgt.

Für den Zugriff auf die ServerStatus-API benötigen Sie Ihre Kundennummer, einen registrierten Usernamen mit Passwort sowie eine IP-Adresse, von der aus Sie auf die ServerStatus-API zugreifen wollen.
Ohne vorherige Registrierung ist ein Zugriff auf die ServerStatus-API nicht möglich!
Usernamen/Passwort erhalten Sie von Ihrem Supportteam. Zugangsdaten für Ihre Kunden können Sie dann selbst über das ServerStatus-API erstellen.

II. Verbindungsaufnahme zum ServerStatus-API

Generell senden Sie einen Anforderungsrequest an die ServerStatus-API. Sie erhalten dann eine Antwort auf Ihren Request von der ServerStatus-API. Sowohl Anforderung als auch Antwort sind im Format XML kodiert.
Zwischen GROSS/kleinschreibung wird unterschieden! Nicht definierte Parameter werden ignoriert (Schreibfehler!).

III. Allgemeiner Aufbau eines Requests:

Anforderungsrequest:

```
<request>
  <content>
    <command>command</command>
    <version>version</version>
    ...
  </content>
  <login>
    <kundennr>kundennummer</kundennr>
    <password>passwort</password>
    <user>username</user>
  </login>
</request>
```

Die Angabe von **command**, **version**, **kundennummer**, **passwort** und **username** sind zwingend erforderlich.

Auf Ihre Anfrage erhalten Sie eine verallgemeinerte Antwort wie folgt:

```
<answer>
  <content>
    <command>command</command>
    ...
  </content>
  <result>
    <errnr>errnr</errnr>
    <error>error<</error>
    <status>status</status>
  </result>
</answer>
```

Bei einer fehlerfreien Ausführung Ihrer Anfrage werden im `<result>`-Abschnitt folgende Daten übermittelt:

errnr: 0
error: Leere Zeichenkette
status: OK

Bei einer fehlerhaften Ausführung Ihrer Anfrage werden im `<result>`-Abschnitt folgende Daten übermittelt:

errnr: von Null verschieden, siehe Liste der Fehlermeldungen.
error: der aufgetretene Fehler im Klartext
status: ERR

Nach der Übermittlung der Antwort auf Ihre Anfrage wird die Verbindung getrennt.

IV. Die einzelnen Kommandos in der Übersicht:

- *CreateUserAccount*
  Erzeugt einen Ihnen untergeordnete Useraccount (i.d.R. Für Ihre Kunden)

- *DeleteUserAccount*
  Löscht einen vorhandenen Useraccount.

- *GetServerList*
  Gibt eine Liste aller Ihrer Server zurück.

- *GetUserAccount*
  Gibt Informationen zu einem Useraccount zurück.

- *GetUserList*
  Gibt eine Liste aller verfügbaren Useraccounts zurück.

- *GetServerProperties*
  Gibt eine Aufstellung aller zu Ihrem Server vorhandenen Daten zurück.

- *Reset*
  Löst einen Reset Ihres Server aus.

- *ResetHistory*
  Gibt eine Liste mit den 10 letzten ausgelösten Resets zurück.

- SetReverseLookup
  Setzt das Reverse Lookup für eine IP-Adresse.

- *SetServerProperties*
  Setzt bestimmte Daten bei Ihrem Server auf andere Werte.

- *Traffic*
  Gibt die für Ihren Server vorhandenen Trafficdaten zurück.

V. CreateUserAccount

Mit dieser Funktion legen Sie einen Ihnen untergeordneten Useraccount an. Die einzelnen <content>-Abschnitte beinhalten folgende Daten:

<internalknr>: Ihre interne Kundennummer für diesen Account.

<ipaddr>: die IP-Adresse, von der aus die Zugriffe auf diesen Account erfolgen. Alternativ dazu ist auch '*' möglich.

<passwd>: das Passwort dieses Accounts.

<username>: der Username dieses Accounts.

<monitoring>: Freischaltung des Monitorings.

<version>: 1.0.1

Alle Angaben sind zwingend erforderlich!

Request:

```
<request>
  <content>
    <command>CreateUserAccount</command>
    <internalknr>INT64</internalknr>
    <ipaddr>IPADDR</ipaddr>
    <passwd>ASCII32</passwd>
    <username>ASCII32</username>
    <monitoring>BOOL</monitoring>
    <version>ASCII</version>
  </content>
  <login>
    <kundennr>INT64</kundennr>
    <password>ASCII32</password>
    <user>ASCII32</user>
  </login>
</request>
```

Nach erfolgreichem Anlegen des Useraccounts werden folgende Daten übermittelt:

Antwort:

```
<answer>
  <content>
    <command>CreateUserAccount</command>
    <internalknr>INT64</internalknr>
    <ipaddr>ASCII15</ipaddr>
    <passwd>ASCII32</passwd>
    <monitoring>BOOL</monitoring>
    <username>ASCII32</username>
  </content>
  <result>
    <errnr>INT64</errnr>
    <error>ASCII</error>
    <status>OK|ERR</status>
  </result>
</answer>
```

Bei einer fehlerhaften Ausführung werden die Daten im <result>-Abschnitt entsprechend ausgefüllt.

Beispiel CreateUserAccount:

Request:

```
<request>
  <content>
    <command>CreateUserAccount</command>
    <internalknr>221</internalknr>
    <ipaddr>1.2.3.4</ipaddr>
    <passwd>sagichnicht</passwd>
    <username>testuser</username>
    <monitoring>1</monitoring>
    <version>1.0.1</version>
  </content>
  <login>
    <kundennr>12345</kundennr>
    <password>sagichnicht</password>
    <user>abcdefgh</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>CreateUserAccount</command>
    <internalknr>221</internalknr>
    <ipaddr>1.2.3.4</ipaddr>
    <passwd>sagichnicht</passwd>
    <username>testuser</username>
    <monitoring>1</monitoring>
  </content>
  <result>
    <errnr>0</errnr>
    <error></error>
    <status>OK</status>
  </result>
</answer>
```

Beispiel für eine fehlerhafte Ausführung:

```
<answer>
  <content>
    <command>CreateUserAccount</command>
  </content>
  <result>
    <errnr>21</errnr>
    <error>User 221 already exists.</error>
    <status>ERR</status>
  </result>
</answer>
```

VI. DeleteUserAccount

Mit dieser Funktion löschen Sie einen Ihnen untergeordneten Useraccount. Die einzelnen `<content>`-Abschnitte beinhalten folgende Daten:

`<internalknr>`: Ihre interne Kundennummer für diesen Account.

`<version>`: 1.0.1

Alle Angaben sind zwingend erforderlich!

Request:

```
<request>
  <content>
    <command>DeleteUserAccount</command>
    <internalknr>INT64</internalknr>
    <version>ASCII</version>
  </content>
  <login>
    <kundennr>INT64</kundennr>
    <password>ASCII32</password>
    <user>ASCII32</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>DeleteUserAccount</command>
    <internalknr>INT64</internalknr>
  </content>
  <result>
    <errnr>INT64</errnr>
    <error>ASCII</error>
    <status>OK|ERR<</status>
  </result>
</answer>
```

Bei einer fehlerhaften Ausführung werden die Daten im `<result>`-Abschnitt entsprechend ausgefüllt.

Beispiel DeleteUserAccount:

Request:

```
<request>
  <content>
    <command>DeleteUserAccount</command>
    <internalknr>221</internalknr>
    <version>1.0.1</version>
  </content>
  <login>
    <kundennr>12345</kundennr>
    <password>sagichnicht</password>
    <user>abcdefg</user>
  </login>
</request>
```

Antwort:

```
<answer>
```

```
  <content>
    <command>DeleteUserAccount</command>
    <internalknr>221</internalknr>
  </content>
  <result>
    <errnr>0</errnr>
    <error></error>
    <status>OK</status>
  </result>
</answer>
```

Beispiel für eine fehlerhafte Ausführung:

```
<answer>
  <content>
    <command>DeleteUserAccount</command>
  </content>
  <result>
    <errnr>22</errnr>
    <error>User 221 don't exists.</error>
    <status>ERR</status>
  </result>
</answer>
```

VII. GetServerList

Mit dieser Funktion können Sie eine Liste aller Server abrufen. Die einzelnen <content>-Abschnitte beinhalten folgende Daten:

<version>: 1.0.1

Die Angabe der Version ist zwingend erforderlich!

Optionale Angaben:

<internalknr>: Ihre interne Kundennummer. Damit wird die Ausgabe der Liste auf diesen Kundenaccount beschränkt,d.h. Die Liste enthält nur die Server für diese Kundennummer.

Request:

```
<request>
  <content>
    <command>GetServerList</command>
    <internalknr>INT64</internalknr>    ***OPTIONAL***
    <version>1.0.1</version>
  </content>
  <login>
    <kundennr>INT64</kundennr>
    <password>ASCII32</password>
    <user>ASCII32</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>GetServerList</command>
    <servercount>INT64</servercount>
    <serverlist>
      <server0>ASCII</server0>
      <server1>ASCII</server1>
      <server2>ASCII</server2>
      ...
    </serverlist>
  </content>
  <result>
    <errnr>INT64</errnr>
    <error>ASCII</error>
    <status>OK|ERR</status>
  </result>
</answer>
```

Bei einer fehlerhaften Ausführung werden die Daten im <result>-Abschnitt entsprechend ausgefüllt.

Beispiel GetServerList:

Request:

```
<request>
 <content>
  <command>GetServerList</command>
  <internalknr>221</internalknr>          ***OPTIONAL***
  <version>1.0.1</version>
```

```
  </content>
  <login>
    <kundennr>123456<</kundennr>
    <password>sagichnicht</password>
    <user>abcdefgh</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>GetServerList</command>
    <servercount>7</servercount>
    <serverlist>
      <server0>KM00001</server0>
      <server1>KM00010</server1>
      <server2>KM00100</server2>
      <server3>KM01000</server3>
      <server4>KM10000</server4>
      <server5>KM20001-01</server5>
      <server6>KM20002-02</server6>
    </serverlist>
  </content>
  <result>
    <errnr>0</errnr>
    <error></error>
    <status>OK</status>
  </result>
</answer>
```

VIII. GetUserAccount

Mit dieser Funktion können Sie Informationen über einen einzelnen, Ihnen unterstellten Useraccounts abfragen. Die einzelnen `<content>`-Abschnitte beinhalten folgende Daten:

`<internalknr>`: Ihre interne Kundennummer für diesen Account.

`<version>`: 1.0.1

Alle Angaben sind zwingend erforderlich!

Request:

```
<request>
  <content>
    <command>GetUserAccount</command>
    <internalknr>INT64</internalknr>
    <version>ASCII</version>
  </content>
  <login>
    <kundennr>INT64</kundennr>
    <password>ASCII32</password>
    <user>ASCII32</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>GetUserAccount</command>
    <internalknr>INT64</internalknr>
    <ipaddr>ASCII15</ipaddr>
    <passwd>ASCII32</passwd>
    <username>ASCII32</username>
    <monitoring>BOOL</monitoring>
    <monitoring_username>ASCII32</monitoring_username>
    <monitoring_url>ASCII</monitoring_url>
  </content>
  <result>
    <errnr>INT64</errnr>
    <error>ASCII</error>
    <status>OK|ERR</status>
  </result>
</answer>
```

Je nach Kontext können im `<result>`-Abschnitt einzelne Daten in der Antwort fehlen, wenn diese zum entsprechenden Zeitpunkt nicht vorhanden sind.

Bei einer fehlerhaften Ausführung werden die Daten im `<result>`-Abschnitt entsprechend ausgefüllt.

Beispiel GetUserAccount:

Request:

```
<request>
  <content>
    <command>GetUserAccount</command>
    <internalknr>221</internalknr>
    <version>1.0.1</version>
  </content>
  <login>
```

```
    <kundennr>123456<</kundennr>
    <password>sagichnicht</password>
    <user>abcdefgh</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>GetUserAccount</command>
    <internalknr>221</internalknr>
    <ipaddr>192.168.111.222</ipaddr>
    <passwd>sagichnicht</passwd>
    <username>testuser</username>
    <monitoring>1</monitoring>
    <monitoring_username>123456_221</monitoring_username>
    <monitoring_url>http://monitoringurl/</monitoring_url>
  </content>
  <result>
    <errnr>0</errnr>
    <error></error>
    <status>OK</status>
  </result>
</answer>
```

Beispiel für eine fehlerhafte Ausführung:

```
<answer>
  <content>
    <command>GetUserAccount</command>
  </content>
  <result>
    <errnr>22</errnr>
    <error>User 221 don't exists.</error>
    <status>ERR</status>
  </result>
</answer>
```

IX. GetUserList

Mit dieser Funktion erhalten Sie eine Liste aller Ihrer Useraccounts. Die einzelnen `<content>`-Abschnitte beinhalten folgende Daten:

`<version>`: 1.0.1

Die Angabe der Version ist zwingend erforderlich!

Request:

```
<request>
  <content>
    <command>GetUserList</command>
    <version>ASCII</version>
  </content>
  <login>
    <kundennr>INT64</kundennr>
    <password>ASCII32</password>
    <user>ASCII32</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>GetUserList</command>
    <usercount>INT64</usercount>
    <userlist>
      <internalknr0>INT64</internalknr0>
      <internalknr1>INT64</internalknr1>
      <internalknr2>INT64</internalknr2>
      ...
    </userlist>
  </content>
     </userlist>
  </content>
  <result>
    <errnr>INT64</errnr>
    <error>ASCII</error>
    <status>OK|ERR</status>
  </result>
</answer>
```

Bei einer fehlerhaften Ausführung werden die Daten im `<result>`-Abschnitt entsprechend ausgefüllt.

Beispiel GetUserList:

Request:

```
<request>
  <content>
    <command>GetUserList</command>
    <version>1.0.1</version>
  </content>
  <login>
    <kundennr>123456<</kundennr>
    <password>sagichnicht</password>
    <user>abcdefgh</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>GetUserList</command>
    <usercount>3</usercount>
    <userlist>
      <internalknr0>221</internalknr0>
      <internalknr1>22387</internalknr1>
      <internalknr2>32145</internalknr2>
    </userlist>
  </content>
  <result>
    <errnr>0</errnr>
    <error></error>
    <status>OK</status>
  </result>
</answer>
```

X. GetServerProperties

Mit dieser Funktion erhalten Sie genauere Informationen zu Ihrem Server. Die einzelnen <content>-Abschnitte beinhalten folgende Daten:

<serverid>: Die ServerID Ihres Servers. Die ServerID können Sie u.a. über die Funktion GetServerList ermitteln.

<version>: 1.0.1

Alle Angaben sind zwingend erforderlich!

Request:

```
<request>
  <content>
    <command>GetServerProperties</command>
    <serverid>ASCII</serverid>
    <version>ASCII</version>
  </content>
  <login>
    <kundennr>INT64</kundennr>
    <password>ASCII32</password>
    <user>ASCII32</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>GetServerProperties</command>
    <serverproperties>
      <autoreset>BOOL<</autoreset>
      <autoresettime>INT64</autoresettime>
      <autoresetcount>INT64</autoresetcount>
      <cpu>ASCII</cpu>
      <extendedipcount>INT64</extendedipcount>
      <extendedipaddr>
         <extendedip0>IPADDR</extendedip0>
         <extendedip1>IPADDR</extendedip1>
         <extendedip2>IPADDR</extendedip2>
         ...
      </extendedipaddr>
      <hdd>ASCII</hdd>
      <lizenz>
        <lizenzcount>INT64</lizenzcount>
        <lizenzextinfo0>ASCII</lizenzextinfo0>
        <lizenztext0>ASCII</lizenztext0>
        <lizenzurl0>ASCII</lizenzurl0>
        <lizenzextinfo1>ASCII</lizenzextinfo1>
        <lizenztext1>ASCII</lizenztext1>
        <lizenzurl1>ASCII</lizenzurl1>
        ...
      </lizenz>
      <macaddr>ASCII</macaddr>
      <managed>BOOL</managed>
      <memory>ASCII</memory>
      <nameserver1>ASCII<</nameserver1>
      <nameserver2>ASCII</nameserver2>
      <operatingsystem>ASCII</operatingsystem>
      <resetter>BOOL</resetter>
      <serverid>ASCII</serverid>
      <serverip1>IPADDR</serverip1>
```

```
      <serverip2>IPADDR</serverip2>
      <servername>ASCII</servername>
      <tarif>ASCII</tarif>
      <internalknr>INT64</internalknr>
    </serverproperties>
  </content>
  <result>
    <errnr>INT64</errnr>
    <error>ASCII</error>
    <status>OK|ERR</status>
  </result>
</answer>
```

Je nach Kontext können im `<result>`-Abschnitt einzelne Daten in der Antwort fehlen, wenn diese zum entsprechenden Zeitpunkt nicht vorhanden sind.

Bei einer fehlerhaften Ausführung werden die Daten im `<result>`-Abschnitt entsprechend ausgefüllt.

Beispiel GetServerProperties:

Request:

```
<request>
  <content>
    <command>GetServerProperties</command>
    <serverid>KM12345<</serverid>
    <version>1.0.1</version>
  </content>
  <login>
    <kundennr>123456<</kundennr>
    <password>sagichnicht</password>
    <user>abcdefgh</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>GetServerProperties</command>
    <serverproperties>
      <autoreset>1</autoreset>
      <autoresettime>15</autoresettime>          *** NUR WENN AUTORESET 1 ***
      <autoresetcount>5</autoresetcount>         *** NUR WENN AUTORESET 1 ***
      <cpu>P4 2.4 GHz</cpu>
      <extendedipcount>2</extendedipcount>
      <extendedipaddr>
         <extendedip0>64.11.43.45</extendedip0>
         <extendedip1>65.11.47.35</extendedip1>
      </extendedipaddr>
      <hdd>2x300GB</hdd>
      <lizenzcount>1</lizenzcount>
      <lizenz>
        <lizenzextinfo0>n/v</lizenzextinfo0>
        <lizenztext0>Plesk 8.0 for Virtuozzo : 10 Domains</lizenztext0>
        <lizenzurl0>http://downloadurl</lizenzurl0>
      </lizenz>
      <macaddr>0123456789FF</macaddr>
      <managed>0</managed>
      <memory>1 GByte</memory>
      <nameserver1>ns.km12345.keymachine.de</nameserver1>
      <nameserver2>ns2.km12345.keymachine.de</nameserver2>
      <operatingsystem>RHES3</operatingsystem>
```

```
      <resetter>0</resetter>
      <serverid>KM12345</serverid>
      <serverip1>87.118.108.80</serverip1>
      <serverip2>87.118.109.80</serverip2>
      <servername>km12345.keymachine.de</servername>
      <tarif>KM3000</tarif>
      <internalknr>765432</internalknr>
    </serverproperties>
  </content>
  <result>
    <errnr>0</errnr>
    <error></error>
    <status>OK</status>
  </result>
</answer>
```

Beispiel für eine fehlerhafte Ausführung:

```
<answer>
  <content>
    <command>GetServerProperties</command>
  </content>
  <result>
    <errnr>9</errnr>
    <error>Server KM12345 not found or not your server.</error>
    <status>ERR</status>
  </result>
</answer>
```

XI. Reset

Mit der Funktion Reset können Sie ein Reset bei Ihrem Server auslösen, sofern dieser über eine Reset-Funktion verfügt. Die einzelnen <content>-Abschnitte beinhalten folgende Daten:

<serverid>: Die ServerID Ihres Servers. Die ServerID können Sie u.a. über die Funktion GetServerList ermitteln.

<version>: 1.0.1

Alle Angaben sind zwingend erforderlich!

Optionale Angaben:

<kundenid>: KundenID, die in das Reset-Log eingetragenn wird (frei wählbar) Übermitteln Sie keine leere KundenID! Unterdrücken Sie statt dessen die Übermittlung.

Request:

```
<request>
  <content>
    <command>Reset</command>
    <kundenid>ASCII</kundenid>          **OPTIONAL**
    <serverid>ASCII</serverid>
    <version>ASCII</version>
  </content>
  <login>
    <kundennr>INT64</kundennr>
    <password>ASCII32</password>
    <user>ASCII32</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>Reset</command>
  </content>  <result>
    <errnr>INT64</errnr>
    <error>ASCII</error>
    <status>OK|ERR</status>
  </result>
</answer>
```

Bei einer fehlerhaften Ausführung werden die Daten im <result>-Abschnitt entsprechend ausgefüllt.

Beispiel Reset:

Request:

```
<request>
  <content>
    <command>Reset</command>
    <kundenid>Alfred5</kundenid>
    <serverid>KM12345</serverid>
    <version>1.0.1</version>
  </content>
  <login>
    <kundennr>123456<</kundennr>
```

```
    <password>sagichnicht</password>
    <user>abcdefgh</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>Reset</command>
  </content>
  <result>
    <errnr>0</errnr>
    <error></error>
    <status>OK</status>
  </result>
</answer>
```

Beispiel für eine fehlerhafte Ausführung:

```
<answer>
  <content>
    <command>Reset</command>
  </content>
  <result>
    <errnr>11</errnr>
    <error>Server KM12345 has no reset.</error>
    <status>ERR</status>
  </result>
</answer>
```

XII. ResetHistory

Mit dieser Funktion können Sie die letzten 10 Einträge des Reset-Logs auslesen. Die einzelnen `<content>`-Abschnitte beinhalten folgende Daten:

`<serverid>`: Die ServerID Ihres Servers. Die ServerID können Sie u.a. über die Funktion GetServerList ermitteln.

`<version>`: 1.0.1

Alle Angaben sind zwingend erforderlich!

Request:

```
<request>
  <content>
    <command>ResetHistory</command>
    <serverid>ASCII</serverid>
    <version>ASCII</version>
  </content>
  <login>
    <kundennr>INT64</kundennr>
    <password>ASCII32</password>
    <user>ASCII32</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>ResetHistory</command>
    <historycount>INT64<</historycount>
    <history>
      <datetime0>TIMESTAMP</datetime0>
      <datetime1>TIMESTAMP</datetime1>
      <datetime2>TIMESTAMP</datetime2>
      ...
      <komment0>ASCII</komment0>
      <komment1>ASCII</komment1>
      <komment2>ASCII</komment2>
      ...
      <user0>ASCII</user0>
      <user1>ASCII</user1>
      <user2>ASCII</user2>
      ...
    </history>
  <result>
    <errnr>INT64</errnr>
    <error>ASCII</error>
    <status>OK|ERR</status>
  </result>
</answer>
```

Bei einer fehlerhaften Ausführung werden die Daten im `<result>`-Abschnitt entsprechend ausgefüllt.

Beispiel ResetHistory:

Request:

```
<request>
  <content>
    <command>ResetHistory</command>
    <serverid>KM12345</serverid>
    <version>1.0.1</version>
  </content>
  <login>
    <kundennr>123456<</kundennr>
    <password>sagichnicht</password>
    <user>abcdefgh</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>ResetHistory</command>
    <historycount>2</historycount>
    <history>
      <datetime0>20070823110614</datetime0>
      <datetime1>20070820162037</datetime1>
      <komment0>via XML API (authenticated).</komment0>
      <komment1>via XML API (authenticated).</komment1>
      <user0>Alfred5</user0>
      <user1>Alfred5</user1>
    </history>
  <result>
    <errnr>0</errnr>
    <error></error>
    <status>OK</status>
  </result>
</answer>
```

XIII. SetReverseLookup

Mit dieser Funktion können Sie das Reverse Lookup einer IP-Adresse verändern. Dabei kann es sich sowohl um eine Server-IP, um eine Extended-IP als auch eine IP aus einem Ihnen als Kundennetz zugewiesenen Netz handeln.

Bei Kundennetzen gelten folgende Regeln:

1. Reservierte IPs (Network, Gateway, Broadcast) können nicht verändert werden.
2. IPs können unabhängig von einer Zuweisung an eine ServerID verändert werden.
3. Von Ihnen angelegten Useraccounts haben nur auf die IPs Zugriff, die explizit einer dem Useraccount zugewiesenen ServerID gehören (Regel 2 aufgehoben).
4. IP-Adressen aus Netzen, die per Classless Delegation an Ihre Nameserver delegiert sind, können nicht verändert werden.
5. IP-Adressen aus Netzen, die per RIPE an Ihre Nameserver delegiert sind, können zwar geändert werden, die Änderung hat jedoch keine Auswirkung.

Einen Reverse Lookup können Sie durch senden von NULL auf den Defaultwert zurücksetzen.

Die einzelnen `<content>`-Abschnitte beinhalten folgende Daten:

`<serverid>`:	Die ServerID Ihres Servers. Die ServerID können Sie u.a. über die Funktion GetServerList ermitteln.
`<ipaddr>`:	Die IP-Adresse, die verändert werden soll.
`<lookup>`:	Wert, auf den das Reverse Lookup gesetzt werden soll.
`<version>`:	1.0.1

Alle Angaben sind zwingend erforderlich!

Request:

```
<request>
  <content>
    <command>SetReverseLookup</command>
    <serverid>ASCII</serverid>
    <ipaddr>IPADDR</ipaddr>
    <lookup>ASCII64</lookup>
    <version>ASCII</version>
  </content>
  <login>
    <kundennr>INT64</kundennr>
    <password>ASCII32</password>
    <user>ASCII32</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>SetReverseLookup</command>
    <serverid>ASCII</serverid>
    <ipaddr>IPADDR</ipaddr>
    <lookup>ASCII64</lookup>
   </content>
   <result>
    <errnr>INT64</errnr>
    <error>ASCII</error>
    <status>OK|ERR</status>
  </result>
</answer>
```

Bei einer fehlerhaften Ausführung werden die Daten im `<result>`-Abschnitt entsprechend ausgefüllt.

XIV. SetServerProperties

Mit dieser Funktion können Sie einzelne Eigenschaften Ihres Servers beeinflussen. Die einzelnen <content>-Abschnitte beinhalten folgende Daten:

<serverid>: Die ServerID Ihres Servers. Die ServerID können Sie u.a. über die Funktion GetServerList ermitteln.

<version>: 1.0.1

Alle Angaben sind zwingend erforderlich!

Optionale Angaben:

<nameserver1>: Der Name des Nameservers der ersten IP-Adresse.
<nameserver2>: Der Name des nameservers der zweiten IP-Adresse.
<servername>: Der Name des Servers.
<internalknr>: Ihre interne Kundennummer des Useraccounts, dem der Server zugeordnet ist. Der Useraccount muß vorhanden sein. Diese Optione steht den von Ihnen angelegten untergeordneten Useraccounts nicht zur Verfügung.

Request:

```
<request>
  <content>
    <command>SetServerProperties</command>
    <serverid>ASCII</serverid>
    <serverproperties>
      <nameserver1>ASCII</nameserver1>          **OPTIONAL**
      <nameserver2>ASCII</nameserver2>          **OPTIONAL**
      <servername>ASCII</servername>            **OPTIONAL**
      <internalknr>INT64</internalknr>          **OPTIONAL**
    </serverproperties>
    <version>ASCII</version>
  </content>
  <login>
    <kundennr>INT64</kundennr>
    <password>ASCII32</password>
    <user>ASCII32</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>SetServerProperties</command>
    <serverid>KM12345</serverid>
    <serverproperties>
      <nameserver1>ASCII</nameserver1>
      <nameserver2>ASCII</nameserver2>
      <servername>ASCII</servername>
      <internalknr>INT64</internalknr>
    </serverproperties>
  </content>
  <result>
    <errnr>INT64</errnr>
    <error>ASCII</error>
    <status>OK|ERR</status>
  </result>
</answer>
```

Bei einer fehlerhaften Ausführung werden die Daten im `<result>`-Abschnitt entsprechend ausgefüllt.

Beispiel SetServerProperties:

Request:

```
<request>
  <content>
    <command>SetServerProperties</command>
    <serverid>KM12345</serverid>
    <serverproperties>
      <nameserver1>ns.km12345.keymachine.de</nameserver1>
      <nameserver2>ns2.km12345.keymachine.de</nameserver2>
      <servername>km12345.keymachine.de</servername>
      <internalknr>1234567</internalknr>
    </serverproperties>
    <version>1.0.1</version>
  </content>
   <login>
    <kundennr>123456<</kundennr>
    <password>sagichnicht</password>
    <user>abcdefgh</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>SetServerProperties</command>
    <serverid>KM12345</serverid>
    <serverproperties>
      <nameserver1>ns.km12345.keymachine.de</nameserver1>
      <nameserver2>ns2.km12345.keymachine.de</nameserver2>
      <servername>km12345.keymachine.de</servername>
      <internalknr>1234567</internalknr>
    </serverproperties>
  </content>
  <result>
    <errnr>0</errnr>
    <error></error>
    <status>OK</status>
  </result>
</answer>
```

Beispiel für eine fehlerhafte Ausführung:

```
<answer>
  <content>
    <command>SetServerProperties</command>
  </content>
  <result>
    <errnr>9</errnr>
    <error>Server KM12345 not found or not your server.</error>
    <status>ERR</status>
  </result>
</answer>
```

XV. Traffic

Mit dieser Funktion werden die für Ihren Server erfassten Trafficdaten zurückgegeben. Die einzelnen <content>-Abschnitte beinhalten folgende Daten:

<serverid>: Die ServerID Ihres Servers. Die ServerID können Sie u.a. über die Funktion GetServerList ermitteln.

<requesttype>: Die Art der Daten, die Sie abrufen möchten. Es gibt 5 mögliche Requesttypen:

- today: Die Trafficgrafik des aktuellen Tages.
- yesterday: Die Trafficgrafik des gestrigen Tages.
- weekly: Die Trafficgrafik der letzten Woche.
- monthly: Die Trafficgrafik des Monats.
- ~~table:~~ ~~Die numerischen Trafficdaten der letzten zwei Monate (in Bytes)~~ (Entfall ab 09.09.2009)
- date: Die numerischen Trafficdaten für das angegebene Datum
- lastslots: Die letzten aktuellen Trafficdaten

<version>: 1.0.1

Alle Angaben sind zwingend erforderlich!

Request:

```
<request>
  <content>
    <command>Traffic</command>
    <requesttype>REQUESTTYPE</requesttype>
    <date>DATE</date>                 *** NUR BEI REQUESTTYPE „date“ ***
    <lastslots>INT64</lastslots>      *** NUR BEI REQUESTTYPE „lastslots“ ***
    <serverid>ASCII</serverid>
    <version>ASCII</version>
  </content>
  <login>
    <kundennr>INT64</kundennr>
    <password>ASCII32</password>
    <user>ASCII32</user>
  </login>
</request>
```

Antwort (je nach Requesttype):

```
<answer>
  <content>
    <command>Traffic</command>
    <requesttype>REQUESTTYPE</requesttype>
------
    <today>
      <png>PNGBITMAP</png>
    </today>
-oder-
    <yesterday>
      <png>PNGBITMAP</png>
    </yesterday>
-oder-
    <weekly>
      <png>PNGBITMAP</png>
    </weekly>
-oder-
    <monthly>
      <png>PNGBITMAP</png>
    </monthly>
```

```
-oder-
    <tablecount>INT64<</tablecount>
    <table>
      <traffic0>INT64</traffic0>
      <date0>DATE</date0>
      <traffic1>INT64</traffic1>
      <date1>DATE</date1>
      <traffic2>INT64</traffic2>
      <date2>DATE</date2>
    </table>
-oder-
    <tablecount>INT64</tablecount>
    <datum>DATE</datum>
    <table>
      <date0>TIME</date0>
      <received0>INT64</received0>
      <transmitted0>INT64</transmitted0>
      <traffic0>INT64</traffic1>
      <date1>TIME</date1>
      <received1>INT64</received1>
      <transmitted1>INT64</transmitted1>
      <traffic1>INT64</traffic2>
      <date2>TIME</date1>
      <received2>INT64</received2>
      <transmitted2>INT64</transmitted2>
      <traffic2>INT64</traffic2>
    </table>
  </content>
  <result>
    <errnr>INT64</errnr>
    <error>ASCII</error>
    <status>OK|ERR</status>
  </result>
</answer>
```

Bei einer fehlerhaften Ausführung werden die Daten im `<result>`-Abschnitt entsprechend ausgefüllt.

Beachten Sie, daß Sie beim Traffictype „lastslots“ nur neue Daten im Raster von fünf Minuten erhalten.
Beachten Sie, das Sie die Tagesgrenze nicht überspringen können.

Beispiel Traffic (today):

Request:

```
<request>
  <content>
    <command>Traffic</command>
    <requesttype>today</requesttype>
    <serverid>KM12345</serverid>
    <version>1.0.1</version>
  </content>  <login>
    <kundennr>123456<</kundennr>
    <password>sagichnicht</password>
    <user>abcdefgh</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>Traffic</command>
    <serverid>KM12345</serverid>
    <requesttype>today</requesttype>
    <today>
      <png>=89PNG=0D=1A=00=00=00=0DIHDR=00=00=02j=00...</png></today>
  </content>
  <result>
    <errnr>0</errnr>
    <error></error>
    <status>OK</status>
  </result>
</answer>
```

Anmerkung für PHP:

Beachten Sie, das möglicherweise alle Antworten im Format

transfer-encoding: chunked

vorliegen. Hinweise dazu finden Sie bei den Skriptbeispielen.

Beispiel Traffic (date):

Request:

```
<request>
  <content>
    <command>Traffic</command>
    <requesttype>date</requesttype>
    <serverid>KM12345</serverid>
    <version>1.0.1</version>
  </content>
  <login>
    <kundennr>123456<</kundennr>
    <password>sagichnicht</password>
    <user>abcdefgh</user>
  </login>
</request>
```

Antwort:

```
<answer>
  <content>
    <command>Traffic<</command>
    <serverid>KM12345</serverid>
    <requesttype>date</requesttype>
    <tablecount>3</tablecount>
    <table>
      <traffic0>750</traffic0>
      <received0>250</received0>
      <transmitted0>500</transmitted0>
      <time0>00:10:00</date0>
      <traffic1>1000</traffic1>
      <received1>400</received1>
      <transmitted1>600</transmitted1>
      <time1>00:15:00</time1>
      <traffic2>2000</traffic2>
      <received2>1800</received2>
      <tramsmitted2>200</transmitted2>
      <time2>00:20:00</time2>
    </table>
  </content>
  <result>
    <errnr>0</errnr>
    <error></error>
    <status>OK</status>
  </result>
</answer>
```

XVI. Server sperren/-freigeben

Mit dieser Funktion können Sie einzelne Server off-und online setzen sowie Informationen zum Onlinestatus abrufen. Bitte beachten Sie, daß möglicherweise administrativ gesperrte Server von Ihnen nicht entsperrte werden können!

`<serverid>`: Die ServerID Ihres Servers. Die ServerID können Sie u.a. über die Funktion GetServerList ermitteln.

`<requesttype>`: Die Art der Aktion, die Sie ausführen möchten. Es gibt 3 mögliche Requesttypen:

- `lock`: Der Server wird gesperrt.
- `unlock`: Der Server wird entsperrt (so möglich).
- `info`: Informationen zur Sperre des Server.

`<version>`: 1.0.0

Alle Angaben sind zwingend erforderlich!

Request:

```
<request>
  <content>
    <command>ServerLock</command>
    <requesttype>REQUESTTYPE</requesttype>
    <version>ASCII</version>
  </content>
  <login>
    <kundennr>INT64</kundennr>
    <password>ASCII32</password>
    <user>ASCII32</user>
  </login>
</request>
```

Antwort (je nach Requesttype):

```
<answer>
  <content>
    <command>ServerLock</command>
    <requesttype>REQUESTTYPE</requesttype>
    <info>(locked|unlocked)</info>  *** NUR BEI REQUESTTYPE „info“ ***
    <authoritative>ASCII</info>     *** NUR BEI REQUESTTYPE „info“ ***
  </content>
  <result>
    <errnr>0</errnr>
    <error></error>
    <status>OK</status>
  </result>
</answer>
```

Beispiel für eine fehlerhafte Ausführung:

```
<answer>
  <content>
    <command>ServerLock</command>
    <requesttype>unlock</requesttype>
  <result>
    <errnr>15</errnr>
    <error>This server can't be unlocked due...</error>
    <status>ERR</status>
  </result>
</answer>
```

XVII. Verwendete Datentypen

OK|ERR:	Zeichenkette OK oder ERR
INT64:	Ganzzahliger Wert (64bit)
ASCII:	Zeichenkette beliebiger Länge
ASCII15	Zeichenkette mit max. 15 Zeichen.
ASCII32:	Zeichenkette mit max. 32 Zeichen.
ASCII64:	Zeichenkette mit max. 64 Zeichen.
BOOL:	Ein Wahrheitswert, i.d.R 0 (falsch) und 1 (wahr).
IPADDR:	IPV4 IPAdresse (Bsp: 192.168.111.222)
TIMESTAMP:	Datum und Zeit im Format YYYYMMDDHHNNSS
DATE:	Datum im Format DD.MM.YYYY
TIME:	Zeit im Format HH:MM:SS
REQUESTTYPE:	Zeichenkette 'today','yesterday','weekly','monthly', 'table' oder 'date'
PNGBITMAP:	PNG-Bitmap kodiert im Format „quoted printable“

XVIII. Beispiel-Implementation

Beispiel Perl:

```
#!/usr/bin/perl
use strict;
use LWP::UserAgent;

#- Setup---
my $KUNDENNR = '12345';
my $USERNAME = 'testuser';
my $PASSWORT = '????????';
my $URL      = 'http://95.169.160.11/';

# Create the request

my $XML = <<ENDXML;
<request>
  <content>
    <command>GetServerList</command>
    <version>1.0.1</version>
  </content>
  <login>
    <kundennr>$KUNDENNR</kundennr>
    <password>$PASSWORT</password>
    <user>$USERNAME</user>
  </login>
</request>
ENDXML

# Sending the request
my $UA = LWP::UserAgent->new;
my $REQ = HTTP::Request->new(POST => $URL);
$REQ->content_type('text/xml');
$REQ->content($XML);
my $RES = $UA->request($REQ);
if (!$RES->is_success) {
  die "Error: " . $RES->status_line . "\n";
}
$XML = $RES->content;

print "$XML\n";
```

Beispiel PHP I (ab PHP3.x):

```
<?php

function MakeRequestGetResponse($host, $port, $path, $requestdata) {
        // connect
        $fp = fsockopen($host, $port);
        //fputs($fp, "POST $path HTTP/1.0\n");
        $request = "POST $path HTTP/1.0\n";
        $request .= "Host: $host\n";
        $request .= "Content-type: text/xml\n";
        $request .= "Content-length: ".strlen($requestdata)."\n\n";
        $request .= $requestdata;

        fwrite($fp, $request);

        // get response
        $response = "";
        while(!feof($fp)) {
                $response .= fgets($fp, 128);
        }
        fclose($fp);

        return $response;
}

$kundennr = 12345;
$user = 12345;
$password = "absolutgeheim";

$requestdata = "<request>
        <content>
                <command>GetServerList</command>
                <version>1.0.1</version>
        </content>
        <login>
                <kundennr>{$kundennr}</kundennr>
                <user>{$user}</user>
                <password>{$password}</password>
        </login>
</request>";

$response = MakeRequestGetResponse("95.169.160.11", 80, "/", $requestdata);
header("Content-type: text/xml");
    echo $response;
?>
```

mit freundlicher Genehmigung durch Markus Goernhardt

Beispiel PHP 2 (PHP5.x):

```
/* Beispielfunktionen zum Abruf der Traffic-PNG-Daten für einen Server per Keyweb-Status-API      */
/* Entwicklung: xtra-media, E. Säwert, Saalfelder Str. 74, 07381 Pößneck                          */
/* Web: http://www.xtra-media.de                                                                  */
/* E-Mail: info@xtra-media.de                                                                     */
/*                                                                                                */
/* Achtung, die Funktionen müssen unter Umständen an bestehende */
/* Umgebungen angepasst werden!      */
/*  */
/* allg. Funktion zur Kontaktaufnahme mit der Keyweb-Server-API */
/* $data_to_send enthält die nötigen Daten für die Keyweb-Status-API */

function ContactRemoteHost($data_to_send) {
        $cfgResetHost = "xml.status.keyweb.de";
        $cfgResetPath = "/";
        $fp = @fsockopen($cfgResetHost, 80);
        if ($fp) {
        fputs($fp, "POST $cfgResetPath HTTP/1.1\r\n");
        fputs($fp, "Host: $cfgResetHost\r\n");
        fputs($fp, "Content-type: application/x-www-form-urlencoded\r\n");
        fputs($fp, "Content-length: ".strlen($data_to_send)."\r\n");
        fputs($fp, "Connection: close\r\n\r\n");
        fputs($fp, $data_to_send);

        $header = array();
        $currentHeader = '';
        while ( '' != ($line=trim(fgets($fp, 1024))) ) {
            if ( false !== ($pos=strpos($line, ':')) ) {
                        $currentHeader = substr($line, 0, $pos);
                        $header[$currentHeader] = trim(substr($line, $pos+1));
                }else{
                        @$header[$currentHeader] .= $line;
```

```
                }
        }

        if (isset($header['Transfer-Encoding']) &&
                    $header['Transfer-Encoding'] == 'chunked') {
                $chunk = hexdec(fgets($fp, 1024));
        }else{
                $chunk = -1;
        }

        $res = '';
        $dbg = 0;
        while($chunk != 0 && !feof($fp) && $dbg++ < 200){
                if ($chunk > 0) {
                        $part = fread($fp, $chunk);
                        $chunk -= strlen($part);
                        $res .= $part;
                        if ($chunk == 0) {
                                if (fgets($fp, 1024) != "\r\n") {
                                        die('Fehler in chunk-decoding');
                                }
                                $chunk = hexdec(fgets($fp, 1024));
                        }
             }else{
                        $res .= fread($fp, 1024);
                }
        }
        fclose($fp);
        }
        return $res;
}



/* Funktion zum Abruf der Traffic-Daten */
/* $traffic_server enthält den internen Servernamen bei Keyweb (z.B KM12345) */
/* $traffic_type kann einer der folgenden Werte sein "today", "yesterday", "weekly", "monthly" */
/* Es wird die Komplette Antwort der keyweb-Status-API zurückgegeben. */

function GetTrafficServer($traffic_server, $traffic_type) {
        $data_to_send = '<request>
                                <content>
                                        <command>Traffic</command>
                                        <requesttype>'.$traffic_type.'</requesttype>
                                        <serverid>'.$traffic_server.'</serverid>
                                        <version>1.0.1</version>
                                </content>
                                <login>
                                        <kundennr>KEYWEB-KUNDENNR</kundennr>
                                        <password>KEYWEB-PASSWORT</password>
                                        <user>KEYWEB-USER</user>
                                </login>
                        </request>';

        $check = ContactRemoteHost($data_to_send);
        if (!ereg("OK", $check) || ereg("ERR", $check)) {
                $out = '<b>FEHLER</b><br>Bei der Trafficabfrage ist ein Fehler aufgetreten.';
        }else{
                $out = $check;
        }
        return $out;
}

/* Abruf und Aufbereitung der Antwort von der Keyweb-Server-API */
/* $traffic_server enthält den internen Servernamen bei Keyweb (z.B KM12345) */
/* $traffic_type kann einer der folgenden Werte sein: */
/* "today", "yesterday", "weekly", "monthly" */

$traffic_xml = GetTrafficServer($traffic_server, $traffic_type);

$traffic_xml =
        "<?xml version=\"1.0\" ?>" .
        substr($traffic_xml, strpos($traffic_xml, "<answer>"));

$traffic_xml =
        substr($traffic_xml, 0, 0 - (strlen(substr($traffic_xml,
        (strpos($traffic_xml, "</answer>"))+9))));

/* Zugriff auf die Bilddaten je nachdem welchen Wert $traffic_type beim */
/* Abruf hatte... */

$xml = @new SimpleXMLElement($traffic_xml);
$image_today = $xml->content[0]->today[0]->png;
// $image_yesterday = $xml->content[0]->yesterday[0]->png;
// $image_weekly = $xml->content[0]->weekly[0]->png;
```

```
// $image_monthly = $xml->content[0]->monthly[0]->png;

/* Ausgabe des Bildes je nachdem welchen Wert $traffic_type beim */
/* Abruf hatte... */

header("Content-type: image/png");
echo quoted_printable_decode($image_today);
// echo quoted_printable_decode($image_yesterday);
// echo quoted_printable_decode($image_weekly);
// echo quoted_printable_decode($image_monthly);
```

XIX. Liste der Fehlermeldungen

Fehlernummer	Bedeutung
1	Request benötigt die Angabe eines Usernamens.
2	Username, Kundennummer oder Password falsch.
3	Login von dieser IP-Adresse nicht zulässig.
4	Der Parameter Version muß angegeben werden.
5	Falsche Version.
6	Unbekanntes Kommando.
7	Parameter „command" benötigt.
8	Verbindung zur Datenbank zeitweise gestört.
9	Server nicht gefunden/Server gehört Ihnen nicht.
10	Parameter „serverid" wird benötigt.
11	Server hat kein Resetsystem.
12	Fehler des Reset-Subsystems.
13	Unbekannter Requesttype.
14	AccountID nicht gefunden.
15	Diese Funktion ist hier nicht zulässig.
16	Parameter „internalknr" wird benötigt.
17	Parameter „username" wird benötigt.
18	Parameter „ipadddr" wird benötigt.
19	Parameter „ipaddr" hat falsches Format.
20	Parameter „internalknr" kann nicht gleich Ihrer Kundennummer sein.
21	Diese AccountID existiert bereits.
22	Diese AccountID existiert nicht.
23	Funktion ist für virtuelle Server nicht verfügbar.
24	Parameter „lookup" benötigt.
25	Ungültiger Domainname
26	Classless Delegation
27	Reservierte IP-Adresse (Network, Gateway, Broadcast etc.)
28	IP-Adresse nicht gefunden oder nicht zugewiesen
29	Parameter „date" erforderlich.
30	Format DATE falsch (DD.MM.YYYY)
255	Allgemeiner Software-Fehler.

XX. Änderungen

1.0.1.6: Neue Funktion ServerLock

1.0.1.3: Funktion GetServerProperties gibt die Internalknr mit zurück.

1.0.1.2: Neuer Traffic-RequestType 'lastslots'

1.0.1.1: Neue Function SetReverseLookup
Neuer Traffic-RequestType 'date'
Beispiel-Implementation PHP (2x)
Funktion SetServerProperties:
reverselookup1 und reverselookup2 entfernt,
siehe SetReverseLookup.