Renée Bäcker

Ticket oder nicht Ticket?

Nicht nur diese Frage kann mit Postmaster-Filtern in OTRS beantwortet werden. Viele Tickets werden bei OTRS wie bei den meisten anderen Ticket-/HelpDesk-Systemen über E-Mails erzeugt. Die E-Mails werden hier von dem so genannten Postmaster abgearbeitet. Bis dann der User die E-Mail als Ticket zu sehen bekommt, werden verschiedene Stufen abgearbeitet. Die E-Mail wird eingelesen (entweder von einem Postfach abgeholt oder über ein Skript herein geschoben), vorgefiltert, ein Ticket wird erzeugt und nachträglich noch einmal gefiltert (siehe Abbildung 1).

In diesem Artikel werde ich verschiedene Wege zeigen, wie solche Filter umgesetzt werden können.

Abbildung 1: Ablauf PostMaster

Die erste Möglichkeit ist das Einrichten von Filtern über die Weboberfläche. Im Adminbereich unter *PostMaster Filter* können die Einstellungen vorgenommen werden. Für die einzelnen Felder, die in einer Mail vorkommen, kann man Bedingungen angeben. Es handelt sich hierbei um Reguläre Ausdrücke. Aus diesem Grund sollte man genau bedenken, wie die Bedingungen für einen Filter aussehen.

Die Bedingungen der einzelnen Felder werden mit "UND" verknüpft. Also nur dann, wenn alle Bedingungen im Filter auf die Mail zutreffen, werden die Aktionen ausgeführt, die im unteren Teil angegeben werden können.

In Abbildung 2 ist ein Beispiel für einen Filter zu sehen, der alle Mails von einem bestimmten Absender in die Junk-Queue verschiebt und automatisch schließt.

Abbildung 2: PostMaster Filter, der SPAM filtert

Stärken und Schwächen dieser Filter

Alle gerade beschriebenen Filter werden ausgeführt, wenn die Tickets schon erzeugt wurden. Man kann hier also keine Mails ablehnen. Der Vorteil liegt aber darin, dass so ein Filter schnell eingerichtet ist. Durch diese Einfachheit in der Bedienung ist aber auch eine gewisse Unflexibilität gegeben. So gibt es keine "ODER"-Verknüpfung der einzelnen Bedingungen und auch eine Umkehrung ist nicht immer zu erreichen.

Wer sich mit Regulären Ausdrücken auskennt, kann mit Negativem Look-Behind etc. helfen, aber auch damit kann man nicht alle Probleme lösen. In Abbildung 3 ist ein Filter zu sehen, der auf Mails von einem bestimmten Absender wartet und der Betreff **kein** "Wichtig" am Anfang des Betreffs enthält.

Ein Problem eher allgemeiner Natur ist, dass nicht alle Mailserver die gleichen Header setzen. Ein Beispiel: Eine Mail wird per *BCC* verschickt. In welchem Mail-Header kann man dann die Empfängeradresse finden? Richtig: Es kommt darauf an! Manchmal in `X-Envelope-To`, bei anderen wiederum in `X-Delivered-To` und es gibt noch weitere Header. Manchmal muss man eine Mail erst analysieren um zu wissen, auf welche Felder man achten muss.

Sollte das Feld noch nicht in den DropDowns zur Auswahl stehen, muss man das Feld in der SysConfig unter *Ticket -> Core::PostMaster* bei dem Konfigurationsparameter *PostmasterX-Header* hinzufügen.

Ein weiteres Problem ist bei einem Kunden aufgetaucht. Dort kam eine Mail von einem Dienstleister an. Wenn man die Ticketansicht aufgemacht hat, hat man den Text "5.00/5 Sterne" gesehen. Diese Tickets sollten nun automatisch verschoben und geschlossen werden. Klingt, als wäre es ein einfacher Filter. Aber es war eine HTML-Mail und im Plain-Text-Teil der Mail stand nur, dass es keinen Plain-Text gibt. Die Filter, die über die Oberfläche erstellt werden, sehen aber nur den Plain-Text einer Mail. Letztendlich musste ein Filter als Perl-Modul geschrieben werden.

Dieses Problem und die anderen Schwächen zeigen, dass man unter Umständen einen anderen Mechanismus für Postmaster-Filter wählen muss: Perl-Module als Filter. Für einfache Aufgaben sind die über die Weboberfläche erstellten Filter aber gut geeignet.

Postmaster-Filter selbst programmieren

Mit den selbstgeschriebenen Modulen kann man nicht nur fertige Tickets noch verändern, man kann auch auf die Mails einwirken, wenn noch gar kein Ticket erstellt wurde. Für diesen Artikel soll ein Postmaster-Filter geschrieben werden,

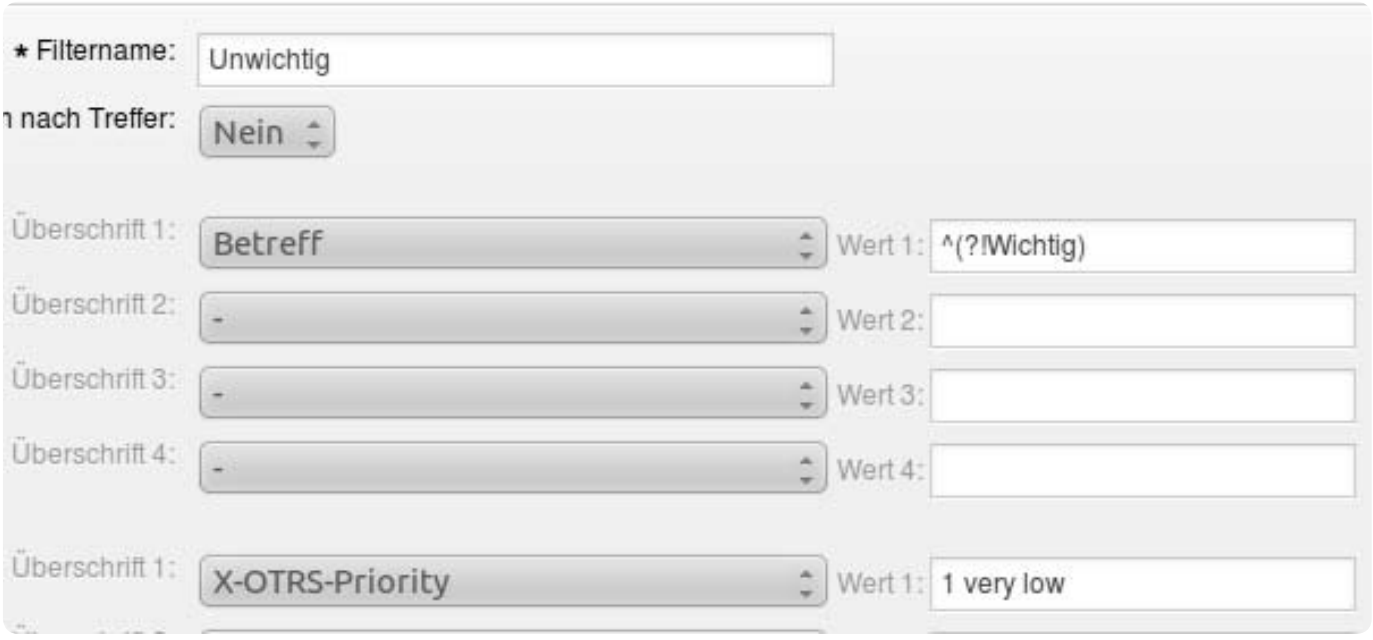

Abbildung 3: Wissen über Reguläre Ausdrücke ist notwendig

der mit Mails umgehen kann, die von einer (fiktiven) Bezahlseite kommen und in denen der Kunde bei der Bezahlung noch ein Kommentar hinterlassen kann. Damit nicht alle Tickets durchgeschaut werden müssen, ob ein Kommentar eingegeben wurde, erstellt der Filter automatisch ein Ticket mit dem Kommentar. Außerdem sollen alle Mails mit den Bezahlinfos in eine Queue "Bezahldienst" geschoben und geschlossen werden.

Zum Parsen der Mails benutzt OTRS verschiedene Module aus dem `MIME::*`-Namensraum und `Mail::Internet` aus der *MailTools*-Distribution.

Gehen wir von folgender Mail aus:

```
Return-Path: <perl@renee-baecker.de>
X-Original-To:
  mailinglisten@renee-baecker.de
Delivered-To: perl@renee
Received: [...]
Message-ID:
  <4F7A126D.7070606@renee-baecker.de>
Date: Mon, 02 Apr 2012 22:56:13 +0200
From: =?ISO-8859-1?Q?Renee_B=E4cker?=
         <perl@renee-baecker.de>
MIME-Version: 1.0
To: mailinglisten@renee-baecker.de
Subject: Bezahldienst: Bezahlung eingegangen
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: 7bit

Der Betrag von 11,53 EUR ist Ihrem Konto
  gutgeschrieben worden.

Der Kunde hat folgende Notiz hinterlassen:

Die Lieferung erfolgte prompt, vielen Dank!
```

Aus dieser Mail erstellt OTRS einen Hash, der an den Postmaster-Filter übergeben wird, dazu gleich mehr.

Ein Postmaster-Filter muss zwei Methoden zur Verfügung stellen: den Konstruktor `new` und die Methode `Run`. In der Methode `Run` müssen die Aktionen implementiert werden, die der Filter leisten soll. Es werden die folgenden Parameter übergeben:

- TicketID
- JobConfig
- GetParam

Sollte die Mail eine Nachricht sein, die zu einem anderen Ticket gehört, steht die ID des Tickets in *TicketID*. Über die SysConfig können weitere Einstellungen zu dem Filter gemacht werden. Existieren solche Einstellungen, werden diese über *JobConfig* an den Postmaster-Filter übergeben.

Die einzelnen Teile aus der Mail stehen in einer Hashreferenz, die über den Parameter *GetParam* in den Filter kommt.

Die Mail von oben wird dann übergeben, wie ins Listing 1 dargestellt.

Dazu kommen noch die ganzen `X-Header`, die über die SysConfig eingestellt werden, für den Beispiel-Filter aber ohne Bedeutung sind.

```
{
  'Content-Type' => 'text/plain; charset=utf-8',
  'X-Sender' => 'perl@renee-baecker.de',
  'X-OTRS-FollowUp-ArticleType' => 'email-external',
  'Resent-From' => '',
  'User-Agent' => 'Mozilla/5.0 (X11; Linux x86_64; rv:11.0) Gecko/20120310 Thunderbird/11.0',
  'Message-Id' => '<4F7A126D.7070606@renee-baecker.de>',
  'From' => "Renee B\x{e4}cker <perl\@renee-baecker.de>",
  'Body' => 'Der Betrag von 11,53 EUR ist Ihrem Konto gutgeschrieben worden.

Der Kunde hat folgende Notiz hinterlassen:

Die Lieferung erfolgte prompt, vielen Dank!
',
   'X-Mailer' => '',
   'ReplyTo' => '',
   'Reply-To' => '',
   'Message-ID' => '<4F7A126D.7070606@renee-baecker.de>',
   'Subject' => 'Bezahldienst: Bezahlung eingegangen',
   'To' => 'mailinglisten@renee-baecker.de',
}
```

Listing 1

Man kann natürlich die geparste Mail mit eigenen Headern anreichern. Diese tauchen nirgends auf, aber sie eignen sich sehr gut dazu, Informationen von einem "Pre"-Filter an einen "Post"-Filter zu übergeben. Das kann man z.B. verwenden wenn in einem Prefilter bestimmt wird, dass ein neues Ticket erstellt werden soll und im Postfilter dann mit anderen Tickets verlinkt werden soll.

Gibt es Anhänge an der Mail, gibt es zusätzlich den Schlüssel *Attachment* in der hashreferenz. Zu diesem Schlüssel wird eine Arrayreferenz übergeben, in der alle Informationen zu den Anhängen zu finden sind:

```
[
  {
      Filename    => $Filename,
      Charset     => $Charset,
      MimeType    => $MimeType,
      ContentType => $ContentType,
      Content     => $Content,
  },
]
```

Doch zurück zum Beispiel. Bevor der Filter geschrieben wird, müssen die Bedingungen festgelegt werden, auf die der Filter reagieren soll. Das sollte möglichst eng gefasst werden, damit der Filter wirklich nur bei den festgelegten Mails greift. Da viel mit Regulären Ausdrücken usw. gearbeitet wird, sollte man den Lauf des Filters möglichst früh abbrechen.

Bei dem Beispiel-Filter muss die Mail von "perl@renee-baecker.de" kommen und der Betreff ist ebenfalls feststehend. Nur wenn Absender und Betreff passen, wird der Text der Mail untersucht. Und dann soll ein neues Ticket aus der Notiz erstellt werden. Und das Ticket aus der Originalmail soll verschoben und geschlossen werden.

```
my $Mail = $Param{GetParam};

return 1 if $Mail->
    {From} !~ /perl\@renee-baecker.de/;
return 1 if $Mail-> {Subject}
    ne 'Bezahldienst: Bezahlung eingegangen';
```

```
my ($Note) = $Mail->{Body} =~ m!Der Kunde hat folgende Notiz hinterlassen:\n(.*)!s;
my $ArticleID = $Self->{TicketObject}->ArticleCreate(
    TicketID         => $TicketID,
    ArticleType      => $Mail->{'X-OTRS-ArticleType'},
    SenderType       => $Mail->{'X-OTRS-SenderType'},
    From             => $Mail->{From},
    ReplyTo          => $Mail->{ReplyTo},
    To               => $Mail->{To},
    Cc               => $Mail->{Cc},
    Subject          => $Mail->{Subject},
    MessageID        => $Mail->{'Message-ID'},
    InReplyTo        => $Mail->{'In-Reply-To'},
    References       => $Mail->{'References'},
    ContentType      => $Mail->{'Content-Type'},
    Body             => $Note,
    UserID           => 1,
    HistoryType      => 'EmailCustomer',
    HistoryComment   => "\%\%$Comment",
    OrigHeader       => $Mail,
    Queue            => $Queue,
);
```

Listing 2

```
<?xml version="1.0" encoding="iso-8859-1"?>
<otrs_config version="1.0" init="Application">
    <ConfigItem Name="PostMaster::PostFilterModule###010-Comments" Required="0" Valid="1">
        <Description Translatable="1">Postmaster filter to handle comments.</Description>
        <Group>PerlServices</Group>
        <SubGroup>Foo::PostMaster</SubGroup>
        <Setting>
            <Hash>
                <Item Key="Module">Kernel::System::PostMaster::Filter::Comments</Item>
            </Hash>
        </Setting>
    </ConfigItem>
</otrs_config>
```

Listing 3

Der Returnwert muss 1 sein, weil sonst im Log die Meldung auftaucht, dass der Filter nicht erfolgreich durchgelaufen sei.

Wenn diese Bedingungen auf die Mail zutreffen, können über *X-OTRS-*-Angaben der Status und die Queue gesetzt werden:

```
$Mail->{'X-OTRS-Queue'} = 'Raw';
$Mail->{'X-OTRS-State'} =
  'closed successful';
```

Und das Ticket muss erzeugt werden:

```
# create new ticket
   my $NewTn    =  $Self->{TicketObject}
                     ->TicketCreateNumber();
   my $TicketID = $Self->{TicketObject}
                     ->TicketCreate(
    TN           => $NewTn,
    Title        => $Mail->{Subject},
    Queue        => $Queue,
    Lock         => 'unlock',
    Priority     => '3 normal',
    State        => 'new',
    CustomerID   => $Mail->{From},
    CustomerUser => $Mail->{From},
    OwnerID      => 1,
    UserID       => 1,
);
if ( !$TicketID ) {
    return;
}
```

Damit sind die Metadaten für das Ticket angelegt. Fehlt noch der Text bzw. der Artikel - siehe Listing 2.

Die meisten Angaben können direkt aus der Originalmail übernommen werden. Einzig der Text enthält nur noch einen Teil der Originalmail.

Bevor der Filter vom Postmaster verwendet wird, muss man diesen über die Konfiguration aktivieren. Hierzu gibt es zwei Wege. Der bequemste Weg ist, unter *Kernel/Config/Files/* eine XML-Datei abzulegen, die folgenden Inhalt hat - siehe Listing 3.

Alternativ kann man in der Datei *Kernel/Config.pm* in der Subroutine `Load` folgenden Eintrag machen:

```
$Self->{'PostMaster::PostFilterModule'}
          ->{'010-Comments'} = {
     Module => 'Kernel::System::PostMaster::
                  Filter::Comments',
 };
```

Hierbei sollte man größeren Wert auf den Namen legen. OTRS sortiert die Schlüssel (Namen) der Filter und führt diese in aufsteigender Reihenfolge aus. Man sollte darauf achten, welchen Namen man vergibt, damit nicht nachfolgende Filter die eigenen Einstellungen überschreiben. Wenn nur Werte gesetzt werden, die von anderen Filtern nicht angefasst werden, ist der Name egal.