4 Agentenberechtigung

Ein Agent möchte auf einem Ticket eine bestimmte Aktion (z.B. Action=AgentTicketNote) ausüben. Ist dies zulässig?

4.1 Permission-Module

Zunächst werden die in (Ticket -> Core::Ticket) Ticket::Permission###* definierten Permission-Module abgearbeitet. Diese sind

1. `Ticket::Permission###1-OwnerCheck`
2. `Ticket::Permission###1-ResponsibleCheck`
3. `Ticket::Permission###2-GroupCheck`

4.1.1 ResponsibleCheck

Es wird überprüft ob der Agent Responsible für das Ticket ist.

Falls

- ja, so werden wegen `Granted=1` die weiteren Module nicht mehr durchlaufen und der Aktionswunsch wird vorläufig bewilligt.
- nein, so werden wegen `Required=0` die weiteren Module noch durchlaufen statt den Aktionswunsch sofort zu untersagen.

4.1.2 OwnerCheck

Identisches verhalten wie in 4.1.1, nur dass diesmal auf den Owner überprüft wird.

4.1.3 GroupCheck

Zunächst wird aktionsabhängig nachgeschaut, welche Permission für die Aktion erforderlich ist. Z.B.

- bei der `Action=AgentTicketNote`
- unter `Ticket -> Frontend::Agent::Ticket::ViewNote`
- der Parameter `Ticket::Frontend::AgentTicketNote###Permission = note`.

Also wird hier die Permission `note` benötigt. Ein zusätzliches Ticket-Lock ist wegen

- `Ticket::Frontend::AgentTicketNote###RequiredLock = Nein`

nicht erforderlich.

Besitzt der Agent diese Permission innerhalb der Queue, in der sich das Ticket derzeit befindet? Falls

- ja, so werden wegen `Granted=1` die weiteren Module nicht mehr durchlaufen und der Aktionswunsch wird vorläufig bewilligt.

- nein, so würde wegen `Required=0` noch weiteren Module durchlaufen werden. Da es aber keine weiteren Module gibt wird der Aktionswunsch untersagen.

4.2 Nachgelagerte ACLs

In (Ticket -> Core::Ticket) `TicketAcl` können ACLs definiert werden die nun, nachdem der Aktionswunsch vorläufig gewährt wurde, den Aktionswunsch noch untersagen können. (Über ACLs können also bereits verwehrte Aktionswünsche nicht nachträglich bewilligt werden.)

Die ACLs werden in alphabetischer Reihenfolge durchlaufen. Trifft eine ACL zu und ist bei dieser `StopAfterMatch` gesetzt, so werden keine weiteren ACLs überprüft.

Es sind statische (siehe http://faq.otrs.org/otrs/public.pl?Action=PublicFAQ&ItemID=68) und dynamische ACLs (siehe 3.1.1) möglich.

4.3 Nachgelagerte Ticket-Sperrung und Owner-Setzen

Falls nun also der Aktionswunsch gewährt wird und dass Ticket gesperrt seien muss (wg. `RequiredLock=Ja` siehe oben) so wird das Ticket gesperrt und, falls noch nicht erfolgt, der Owner auf den Agenten gesetzt.

4.4 Mögliche Permissions

Die möglichen Permissions sind unter (Framework -> Core) `System::Permission` definiert. Diese sind:

Bezeichnung	Recht, um	Action
`ro`	ein Ticket zu lesen	AgentTicketZoom
`move`	ein Ticket aus einer Queue dieser Gruppe zu schieben	AgentTicketMove
`move_into`	ein Ticket in eine Queue dieser Gruppe zu schieben	AgentTicketMove
`create`	ein Ticket in einer Queue dieser Gruppe zu erzeugen	AgentTicketPhone, AgentTicketEmail
`priority`	die Priorität zu ändern	AgentTicketPriority
`forward`	ein Ticket weiterzuleiten	AgentTicketForward
`lock`	ein Ticket zu sperren	AgentTicketLock
`owner`	den Besitzer des Tickets zu ändern	AgentTicketOwner
`responsible`	den Verantwortlichen des Tickets zu ändern	AgentTicketResponsible
`phone`	einen Telefonanruf auf dem Ticket zu protokollieren	AgentTicketPhoneOutbound
`customer`	den Kunden des Tickets zu ändern	AgentTicketCustomer
`freetext`	die freien Textfelder zu ändern	AgentTicketFreeText
`note`	eine Notiz zu hinterlegen	AgentTicketNote
`pending`	das Ticket aus warten zu setzen	AgentTicketPending
`compose`	eine Mailantwort an den Kunden zu verfassen	AgentTicketCompose
`close`	das Ticket zu schließen	AgentTicketClose
`rw`	sämtliche Rechte auf dem Ticket zu erhalten	

Tabelle 1: Permissions

Wie bereits oben erwähnt sind die Permissions, die für die einzelnen Aktionswünsche erforderlich sind, innerhalb der Aktions-Konfiguration definiert, z.B.

```
Ticket::Frontend::AgentTicketNote###Permission = note
```

4.5 Permissions vergeben

Die Permissions auf eine Gruppe kann für

- Agenten und
- Rollen

vergeben werden, d.h. dass z.B. ein Agent oder eine Rolle „ro“-Permission auf der Gruppe xyz hat. Jede Queue ist genau einer Gruppe zugeordnet. Ein Ticket ist zu einem Zeitpunkt genau einer Queue zugeordnet. Möchte also ein Agent ein Ticket lesen, so wird

1. die derzeitige Queue des Tickets ermittelt

2. die Gruppe dieser Queue ermittelt

3. überprüft ob der Agent oder eine Rolle, der der Agent zugehörig ist, die benötigte „ro“-Permission hat.

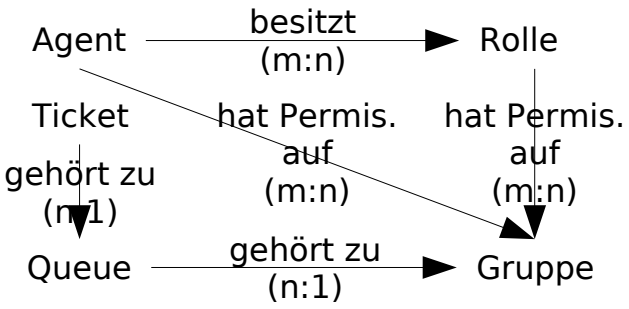

Abbildung 1: Berechtigungssystem